Summer 7-28-2011

Wireless Embedded Smart Cameras: Performance Analysis and their Application to Fall Detection for Eldercare.

Alvaro Pinto
University of Nebraska - Lincoln

WIRELESS EMBEDDED SMART CAMERAS: PERFORMANCE ANALYSIS AND THEIR APPLICATION TO FALL DETECTION FOR ELDERCARE

by

Alvaro Pinto

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Electrical Engineering

Under the Supervision of Professor Senem Velipasalar

Lincoln, Nebraska

August, 2011

WIRELESS EMBEDDED SMART CAMERAS: PERFORMANCE ANALYSIS AND THEIR APPLICATION TO FALL DETECTION FOR ELDERCARE

Alvaro Pinto, M. S.

University of Nebraska, 2011

Adviser: Senem Velipasalar

In the last decade, an alliance between Wireless Sensor Networks and Embedded Smart Cameras (Wireless Embedded Smart Camera Networks) has received a lot of attention from academia and industry, since there exist many unsolved research problems, wireless embedded platforms are small in size and easy to deploy, and they offer a multitude of attractive applications. A Wireless Embedded Smart Camera (WESC) is a standalone unit that combines sensing, processing and communication on a single embedded platform. They provide a lot of flexibility in terms of the number and placement of the cameras. They can be used for embedded single unit applications, or can be networked for multi-camera applications.

We first analyze three different operation scenarios for a wireless vision sensor network wherein different levels of local processing is performed, and thus different amounts of data is transferred in the network. A detailed quantitative comparison of these operation scenarios are presented in terms of energy consumption and latency. This quantitative analysis provides the motivation for, and emphasizes the importance of performing high-level local processing and decision making at the embedded sensor level and need for peer-to-peer communication solutions for wireless multimedia sensor networks.

Then, we present a multi-camera tracking application wherein the amount of data exchanged between cameras has an effect on the tracking accuracy, the en-

ergy consumption of the camera nodes and the latency. We analyzed the tradeoff for these important parameters using different scenarios. Two main scenarios are tested: overlapping and non-overlapping camera setups when different-sized data packets are transferred in a wireless manner.

Finally, we present a lightweight algorithm to perform fall detection for elder-care using wearable embedded smart cameras. This method uses image analysis for a single-unit application. It has low computational cost and fast response time. The proposed lightweight fall detection algorithm uses spatial and temporal derivatives, and a moving sum approach. The experimental results show the success of the algorithm in detecting actual falls, and differentiating falls from most of the regular daily actions. All the experiments have been performed with an actual wireless embedded smart camera(s). We employed CITRIC motes in our experiments.

DEDICATION

This thesis is dedicated to my mother and the greatest influence on my life, Dr. Consuelo Pinto Albuja.

ACKNOWLEDGMENTS

I would like to express my sincere thanks to Dr. Senem Velipasalar for giving me an opportunity to pursue a Master of Science Degree and for her support, guidance and patience throughout this project. I would also like to express my thanks to my committee members: Dr. M. Cenk Gursoy and Dr. Mehmet Can Vuran for their invaluable guidance and support.

I would also like to express my gratitude to all my friends especially, Mauricio Casares, Samy Akin and Fabio Pariggi for their extended support. With their help in countless ways it was possible for me to complete this research project.

Contents

List of Figures

Chapter 1

Introduction

In the last decade, an alliance between wireless sensor networks and embedded smart cameras have received a lot of attention in both academia and industry, since there exist many unsolved research challenges. Also, wireless embedded platforms are small in size, and easy to deploy, and they offer a multitude of attractive applications.

Wireless Embedded Smart Cameras (WESC) are standalone units that combine sensing, processing and communication on a single embedded platform. They provide a lot of flexibility in terms of the number and placement of the cameras. They can be used for embedded single-unit applications, or can be networked for multi-camera applications. The embedded smart cameras must be able to perform on-board processing with a limited storage facility, be able to make some subsequent intelligent decisions and then provide successful transmission over the wireless link using limited energy resources such as batteries.

Research on experimental testbeds and development of real-time algorithms allows researches in the field evaluate their implementations at application-level and network-level.

The first part of this work deals with networked applications while in the second part a single-unit application is developed and deployed on an actual embedded smart camera.

1.1 Networked Applications

An agreement among the researchers has to be reach about the characteristics that experimental platforms must posses. Thus, tools for collection and statistical analysis of experimental data, and techniques, for ensuring that the new algorithms designed for these specific platforms are accurate, must be designed and deployed.

Experiments with wireless embedded cameras are complex and hard to repeat, most of them are use simulations , but in this specific case are unable to model many essential characteristics of real systems. This gap must be solve. Thus, will be of fundamental importance to obtain experimental validation [1].

Different software and hardware approaches have been proposed to minimize the energy consumption in wireless embedded smart camera networks. The majority of the work has concluded that by reducing the energy consumption due to communication is a very important step towards this goal [2].

Even if energy efficiency is taken into account, there is still not an adequate framework for the performance evaluation of wireless embedded smart camera networks. In many applications of wireless embedded smart camera networks, only certain events are of interest to the observer, and the amount of data transferred has an effect on the performance. Therefore, a main challenge in wireless embedded smart cameras networks is to find a tradeoff between performance and energy consumption.

When wireless embedded cameras are used to capture and transfer image and video data, the problems of limited energy and bandwidth become even more pronounced. We present in Chapter 3 that message traffic should be decreased to reduce the communication cost. In many applications, the interest is to detect composite and semantically higher-level events based on information from multiple sensors. Rather than sending all the information to the sinks and performing composite event detection at the sinks or control-center, it is much more efficient to push the detection of semantically high-level events within the network, and perform composite event detection in a peer-to-peer and energy-efficient manner across embedded smart cameras. In this chapter, three different operation scenarios are analyzed for a wireless vision sensor network. A detailed quantitative comparison of these operation scenarios are presented in terms of energy consumption and latency. This quantitative analysis provides the motivation for, and emphasizes (1) the importance of performing high-level local processing and decision making at the embedded sensor level and (2) need for peer-to-peer communication solutions for wireless multimedia sensor networks.

Wireless embedded smart cameras provide flexibility in camera deployment in terms of the locations and number of the cameras. However, these battery-powered embedded vision sensors have very limited energy, memory, and processing power. Energy consumption and latency are two major concerns in wireless embedded camera networks. In multi-camera tracking applications, the amount of data exchanged between cameras has an effect on the tracking accuracy, the energy consumption of the camera nodes and the latency. In chapter 4 we present an analysis of the tradeoff for these important parameters using different scenarios. Two main scenarios are tested: overlapping and non-overlapping camera setups when different-sized data packets are transferred in a wireless manner. The ex-

periments have been performed with an actual wireless embedded smart camera network employing CITRIC motes, and performing tracking of objects.

1.2 Single-Unit Application

Embedded smart cameras are useful in a variety of scenarios: such as surveillance, military applications, medical applications, etc. We have developed an algorithm to perform fall detection for eldercare using wearable wireless embedded smart cameras. The proposed algorithm has low computational cost and fast response. The lightweight fall detection algorithm uses a spatial and temporal derivatives and a moving sum approach to estimate the significant changes in the movement of vertical and horizontal edges. The moving sum allows to recover the underlying trend of the time series. In other words, the abrupt movements which are related to falls can be detected either in the horizontal and vertical direction. Finally, we detect the fall based on empirical thresholds, obtained from different experimental setups. The motivation and the preliminary work are presented in chapter 5 and the lightweight fall detection algorithm is described in chapter 6

The remainder of this thesis is organized as follows: we provide an overview of wireless embedded smart camera networks in chapter 2, focusing on topics that are most relevant for the work presented in this thesis. In chapter 3, we present a detailed quantitative comparison of three scenarios in terms of the energy consumption and latency when the goal is detecting a composite and semantically high-level event. In addition to providing motivation for and emphasizing the importance of pushing the high-level decision making to the sensor level, this analysis gives quantitative results in terms of savings in energy. Chapter 4 provides a detailed quantitative analysis of the accuracy-latency-energy tradeoff for

overlapping and non-overlapping camera setups when different-sized data packets are transferred in a wireless manner. Chapter 5 presents the motivation for and initial work towards developing an algorithm to perform fall detection for eldercare using wearable wireless embedded cameras. The theory and development of a lightweight fall detection algorithm are described, and the experimental results obtained with actual embedded cameras are presented in Chapter 6. Finally, the thesis is concluded in Chapter 7.

Chapter 2

Background

In this chapter, an overview of Wireless Embedded Smart Camera Networks is presented. We present the most important characteristics, design challenges and metrics for the evaluation of the performance of this challenging field. The overview begins with the development of the wireless embedded smart cameras, the core features and challenges face for this field, then a quick review of its architecture. A review of the middleware use is also presented, then a summary of WESCN classification and some examples are presented. Finally, applications for this growing and challenging field.

2.1 Develop of Wireless Embedded Smart Camera Networks

Advances and mature technologies in wireless communication, visual sensor devices, and digital signal processing have enable the development of low cost and low-power Wireless Embedded Smart Camera Networks (WESCN), which have recently emerged for a variety of applications, e.g. surveillance, traffic monitoring, etc. The notion of WESCN can be understood as the convergence of wireless

multimedia sensor networks (WMSN) and distributed smart cameras [1].

Distributed Smart Cameras Computer vision plays an important role in many applications ranging from surveillance, industrial robotics to smart environments [3], [1]. Research in wireless vision networks has been focused on two different assumptions, first is sending all data to the central base station or sink without local processing, second approach is based on conducting all processing locally at the camera.

For the last two decades, the trend towards the implementation of advanced computer vision methods on embedded system have increased substantially. Yet, deployment of advanced vision methods on embedded platforms is challenging, since these platforms often provide only limited resources such as computing performance, memory and power [4]. The solution which has been proposed to tackle this problems are the so called ”smart cameras”.

Since, Wolf [5] and his group present one the first prototypes of its kind many advances have been done in this field. A complete review of the evolution of Smart Cameras is provide in [6]. Bramberguer et al.[6] states that Smart Cameras are part of the third generation of systems. Those systems range from analog equipment, which use close circuit television cameras to digital smart cameras, which analyze and scene and report items to the user [7].

A more formal definition of smart cameras is provided in [6],[5],[7], [8] and [9]. As a result, we can conclude that smart cameras are real-time embedded systems which capture high level descriptors. They are equipped with high-performance, on-board computing and communication infrastructure.

When several cameras are united to cover big areas and solve problems faced by single camera setups; for instance, occlusion. We created a distributed camera. Additionally, if distributed algorithms are used to execute multi-camera tasks; thus, we have a distributed smart camera [7].

The incentive for utilizing distributed processing is to achieve real-time vision and provide scalability for the developing of more complex vision algorithms.

Wireless Multimedia Sensor Networks Wireless Multimedia Sensor Networks (WMSN) is an emerging field, derived from wireless sensor networks. WSN respond to scalar information obtained from various internal sensors such us temperature, light, humidity, pressure, etc. WMSN interconnects autonomous devices for capturing and processing video and audio sensory information [3]. WMSN is a networking paradigm which allows retrieve video streams, still images and generic sensing data from the environment [10]. Therefore, a WMSN will have the ability to store, process in real time, correlate, fuse, transmit and receive multimedia information by using a wireless channel [3], [10], [1].

WMSNs have some novel features, but those features bring more requirements regarding computing power and communication bandwidth than those typically used in wireless sensor networks applications. Therefore, power management is an even more critical issue.

As a result, with the evolution and fusion of technologies from WMSN and distributed smart camera. WESCN are emerging as useful and powerful systems [11]. The new wireless embedded camera prototypes concatenated the best features of both systems. In order to have real-time performance, the embedded smart cam-

eras must be able to perform on-board processing with a limited storage facility, be able to make some subsequent intelligent decisions and then provide successful transmission over the wireless link [12] using limited resources such as batteries.

Wireless networks, however, introduce new constraints of limited bandwidth, computation, and power. Such camera based networks could easily be installed in out-doors areas where there is a limited availability of power, where access is difficult and it is inconvenient to modify the locations of the nodes or frequently change the batteries [2].

However, this promising field raise research problems in the two fields that must be addressed simultaneously. In the next section, an outline of the main features and constraints that WESCN face are provided.

2.2 Core Features and Challenges of Wireless Embedded Smart Cameras

With on-board image processing and analysis capabilities, cameras not only open new possibilities but also raise new challenges. Some of the main characteristics and requirements of WESCN are listed next.

- **Resource Requirements**.

  The bottleneck of WECN mainly resides in the limited energy for each sensing node to perform image sensing, data processing and communication for a long period of time. Fundamental physical limits (Moore's Law) dictate that, as electronics become ever more efficient, communication will dominate a node's energy consumption [13]. For important events, images still might be stream allowing operators to visually evaluate the situation [9].

The large amount of data generated by a vision sensor node is limited by the energy consumption, which is proportional to their energy required for data processing and data transmission over the wireless medium [7] [14].

- **Local Processing**.

  On-board processing, hierarchical collaboration, and domain knowledge is imperative to reduce image data to short descriptive vectors for objects or events [8] . Attention must be paid to the hardware/software design strategy to meet both processing and power requirements. The ideal system has to reduce as much data as possible, as early as possible.

- **Scalability**.

  The nature of WSCN allow us to use them in a spatially distributed manner, this characteristic along with distributed processing strategies provided scalability for the complexity of vision algorithms. [14].

- **Real-time performance**.

  Many applications in WESCN require real-time data from the camera mote. In other words, there are strict maximum delay boundaries that must be achieve. Two main aspects play an important role in WESCN's real-time performance. The first is the time which the embedded camera takes to process the information. Usually, the camera-mote needs to be capable of performing complex image processing such as tracking, which needs a lot of processing. High processing requirement is increased for an increased image size [15] [16]. The embedded processor at the camera-mote dictates the processing speed. Constrained by limited energy resources as well as by allowable delays, most camera-motes have processors that support only

light-weight processing algorithms . Wireless embedded cameras need flexible memory models to meet requirements such as scalable frame buffers to cope with increasing image sensor resolutions. As the smart camera may integrate different types of processors, the memory system should support potentially complex processing pipeline and parallelism in order to meet the applications real-time requirements. For single chip smart cameras, care needs to be taken at design stage to conserve memory [17].

The other aspect that introduces delay to the system is the round trip-delay between motes i.e. the time that takes to make a decision from the source (camera) to the user (sink) and vice-versa. The latency introduced depends on the maximum data rate given by the channel bandwidth which depends on the networking standard employed [16].

- **Precise Location and Orientation Information**.

  In distributed smart cameras, most of the image processing algorithms require information about the locations of the camera-nodes as well as information about the cameras orientations. This information can be obtained through a camera calibration process, which retrieves information on the cameras' intrinsic and extrinsic parameters.

- **Distributed Algorithms.**

  Traditional multi-camera computer vision algorithms assume that the information from all cameras is losslessly communicated to a central processor that solves the problem. This assumption is unrealistic for emerging wireless smart camera networks. The distributed algorithms for WESCN not only are well-suited to ad-hoc wireless networks, have no single point of failure, make fairer use of underyling communication links and computational

resources, but also are more robust and scalable than centralized algorithms [18] .

Two main approaches which are used for distributed algorithms are: consensus and coordination algorithms [19, 20].

Many distributed estimation problems arising from visual sensor networks can be viewed as special types of consensus problems. That is, all nodes in the network should come to an agreement about global parameters describing the network, or objects moving in it, without communicating all sensor data to a centralized location. For instance, all the cameras in the network might need to agree on the estimated position of a central landmark. These techniques are attractive because require communications only between neighbors and converge under mild network connectivity requirements. Unfortunately, these algorithms are designed for scalar, Euclidean quantities [18] .

Distributed coordination aims at achieving collective group behavior through local interaction, it can be seem as a token-passing system, where each node maintains its own internal state and exchanges signals wit other nodes to affect both its own state an that the other nodes. Coordination algorithms in WESCN consider that the interaction among different cameras may be dynamic due to unreliable communication, limited communication/sensing range, and/or sensing with a limited field of view [19].

Radke in [18] presents an extensive overview of distributed algorithms that solve computer vision problems in a distributed manner considering the implementation on a visual sensor network and emphasize simple tracking rather than the estimation of more subtle object/environment characteristics.

In [19] Rinner et al. highlight the necessity to do more research, in order to combined these approaches in to a unified algorithmic framework.

- **Heterogeneity**

  The use of diverse kind of cameras with different resolutions, different communications systems can be consider. We can used scalar systems to activated the camreas and saved power while the camera is not detecting an object in its FoV.

- **Privacy and Security**

  With the evolution of WESCN the deployment of cameras is no longer limited to public places. An example where cameras are installed in private environments is assisted living [21, 22]; where WESCN are use to monitor and study the behavior of elderly people; for instance, falls. In WECN potential security issues are often overlooked. The increasing amount of software running on the cameras turns them into attractive targets for attackers. With the smart camera based approach , only the camera itself remains as a point of attack; consequently, the protection of camera devices and delivered data is of critical importance [23].

  Vision end-user applications requires high privacy and confidentiality; inboard processing gives WESCN an advantage in terms on privacy and security, but still many work have to be done in terms on security of the data which is closely related to sensor networks. Thus, a wide range of mechanism and protocols should be included in the design WESCN.

  A more detailed discussion on the security and privacy issues can be found in [24] and [3]

- **Service Orientation and User Interaction.**

  One factor that is usually forgotten is that these system are going to be used by personal, which is not trained in cameras, then the system must be design for targeted consumer applications.

  As the field of human-computer interaction evolves to exploit smart cameras, new problems of human-system and human-environment interaction will arise. Some methods may find new applications as software add-ons for digital cameras, cellular phones, and personal data assistants [25].

- **Quality of service (QoS)**

  Quality of service is an overused term with multiple meanings and perspectives from different research and technical communities. In WESNs, we can view QoS from two perspectives: application specific and network. The former refers to QoS parameters specific to the application, such as sensor node measurement, deployment, and coverage and number of active sensor nodes. The latter refers to how the supporting communication network can meet application needs while efficiently using network resources such as bandwidth and power consumption.

  Traditional QoS mechanisms used in wired networks are not adequate for WESCNs because of constraints such as resource limitations and dynamic topology. Therefore, middleware should provide new mechanisms to maintain QoS over an extended period and even adjust itself when the required QoS and the state of the application changes. Middleware should be designed based on trade-offs among performance metrics such as network capacity or throughput, data delivery delay, and energy consumption [26].

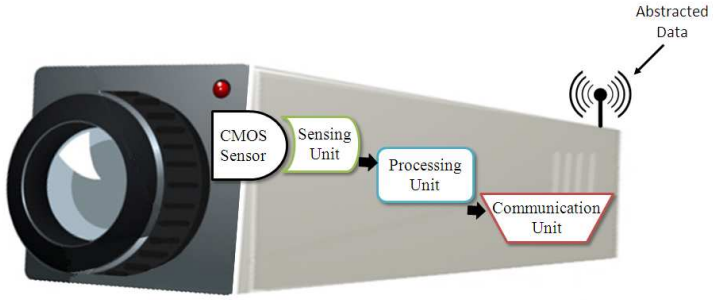


Figure 2.1: General architecture for wireless embedded smart cameras

Data-delivery includes snapshot-multimedia events that contains event triggered observations obtained in short time period, also the variable channel capacity might be bursty and affect QoS.

Significant research is still necessary to achieve the ultimate goal of having available ubiquitous, adaptive, secure and autonomous camera networks.

2.3 Wireless Embedded Smart Architecture

Generally, the hardware of wireless smart cameras can be divided into three main modules as is shown in figure 2.1 :

Data acquisition module .

Essentially composed of an image sensing device. However, other types of data acquisition devices can be integrated in order to obtain supplementary information about the scene or the environment.

Data processing module .

The application-specic information processing (ASIP) is performed by this module. The obtained results can be sent to an external host or network, trigger an event, and/or be used for a feedback loop to control the data acquisition module.

Communication interface module .

Connects the smart camera to the external world (host or network). [25]

2.3.1 Sensor Modules

The main component of the image capture unit or camera front end is essentially a solid state image sensor. The image sensor is the eyes of the smart camera, or any camera. Nowadays, there are mainly two solid state image sensors to choose from, CCD and CMOS. The main technical parameters of an image sensor include resolution, frame rate, scan type, light sensitivity, and noise level. The advent of CMOS image sensors is arguably the most significant single factor that has contributed to the popularity and proliferation of smart cameras across many application areas. The main advantages of CMOS sensors, compared to CCD, include smaller size, cheaper manufacturing cost, lower power consumption, the ability to build a camera-on-a-chip, the ability to integrate intelligent processing circuits onto the sensor chip, and significantly simplified camera system design. The ability of integrating on-chip image processing logic and circuitry makes it possible to create single-chip smart cameras or smart sensors. These very small form factor cameras can be very useful where physical space or power consumption is very restricted [17]. Moreover, the sensor module performs conversion to to digital signals and basic image enhancement task such us contrast white bal-

ance, etc. All those task and others can be usually be controlled [20]. Dynamic range is still one of the key aspects where CMOS image sensors lag behind CCD. Improvement in this area can lead to more low-cost smart cameras using CMOS image sensors for machine vision, surveillance applications, etc. [17].

Additionally, to traditional image sensors; there are many special and usually more expensive sensor that can be used in WESCN such us thermal-imaging, multi-spectral and neuromorphic or "silicon-retina" . Finally, is important to mention that a stand-alone unit can have more than on sensor attached to it [20].

2.3.2 Processing Module

Embedded image processors are the brains of the smart cameras. Nowadays, embedded smart cameras use powerful embedded microprocessors running up to 520 Mhz. A range of processors types can be used.

- *General-purpose processor(GPP).*

  Also known as a embedded microprocessors, Examples are Intel Pentium, Celeron, etc. They are relatively cheap and exible in use; however, they are not ideal for real-time image processing tasks and the power consumption is too high for most applications.

- *Digital signal processors(DSP).*

  They generally provide higher performance for image processing algorithms. However, they are ussually not fine-tuned for processing of this specific type.

- *Media processors*.

  Media processors can be thought of as a special class of DSPs. They provide reasonable cost-effective and flexibility. They typically have a high-end

DSP and VLSI architectures, married on-chip with some typical multimedia peripherals such as video ports, networking support, and other fast data ports. Examples are Trimedia (NXP), DXM64x (TI), Backfin(ADI) and BSP (Equator).

- *FPGA (Field Programmable Gate Array) with embedded processors*

  The FPGA has recently emerged as a very good hardware platform candidate for embedded vision systems such as smart cameras, especially in academia and the research environment. One of the most important advantages of the FPGA is the ability to exploit the inherently parallel nature of many vision algorithms. Many FPGA manufacturers embed microprocessors into FPGA, making them more versatile and powerful. However, FPGAs power consumption is relatively high, and even if design methodologies and development environments exist, FPGA-based solutions require more development time and expertise than CPU-based solutions (DSP, microcontroller, etc.) [25]

- *Image/vision processors*.

  The nature of the processing images where we can apply algorithms like convolution which work in pixels or limited neighborhood of the current pixels; give us and advantage to treat millions of pixels with identical treatment. Some DSP cores, which present a dedicated architecture and some particular hardware structures in order to optimize the execution of arithmetical operations, like MAC (multiply-accumulate) and Single-instruction multiple data (SIMD) units are used. An example of this approach is a Xetal-II processor from NXP. [19].

- *Hybrid processors/ System on Chip (SoC)*.

Smart cameras deals with data communication and possible with controlling external devices. Thus a combination of processing cores for different tasks seems highly suitable for embedded smart cameras. In terms of performance and power consumption, application-specific integrated circuits (ASICs) can be considered as being the ideal choice. Of course, developing a dedicated SoC (system on chip) for a given application allows to fully exploit the silicon, implementing custom architectures can optimize power consumption. However, the development costs for such devices can be prohibitive, making this solution interesting only for consumer products (i.e., a production volume of several thousand units).[25, 19]

Consequently, choosing and designing a processing module that consumes low power depends on the application; additionally, the amount of available memory and the choice between fix-point and floating-point processors posses a high importance.[20, 17]

When choosing embedded processors, the choice of an operating system and the complexity of application along with mature development tools for the chip have to be considered jointly. More details about this topic are consider in section 2.4

2.3.3 Communication Modules

In a wireless sensor network, the communication method varies depending on the application either at the medical, industrial or scientific. One of the most widely used communication protocols is the ZigBee protocol, which is a technology composed of a set of specifications designed for wireless sensor networks. This system is characterized by the conditional type of communication, its mean, which not

require a high volume of information (just over a few kilobits per second) and also have a limited walking distance.

ZigBee, also known as IEEE 802.15.4, can operate at three different frequency bands. This protocol is divided into layers according to the OSI model, where each layer has a specific function depending on the application of our network. The physical layer and access control to the medium (MAC) are standardized by the IEEE 802.15 (WPAN) which is a working group under the name of 802.15.4, where higher layers are specified by ZigBee Alliance. Some characteristics of the layers are given below:

Physical Layer ZigBee / IEEE 802.15.4. The IEEE 802.15.4 physical layer supports unlicensed industrial, scientific and medical radio frequency bands including 868 MHz, 915 MHz and 2.4 GHz.

MAC Layer ZigBee / IEEE 802.15.4. At the MAC layer,there are 2 options to access the medium: Beacon-based (based on orientation) and non-beacon (based on non-guidance). In a non-oriented, there is no time for synchronization between ZigBee devices. The Devices can assess to the channel using (CSMA / CA).

Protocol to the network layer / IEEE 802.15.4. ZigBee got a multi-hop routing and help the capabilities designed as an integral part of the system. This function is implemented within the network layer. [27]

2.4 Embedded Middleware for Embedded Smart Cameras

Middelware is system-level software that resides between the applications and the underlaying operating systems, network protocol stacks, and hardware. The primary function is bridge the gap between application programs and the lower level hardware and software infrastructure in order to make it easier and more cost-effective to develop distributed systems[19].

Middleware implementations are usually extensive, they not only have to run on different hardware platforms, support various communication channels and protocols, but they also have to bridge applications running on different platforms and possibly in different programming languages into a common distributed system; exploiting such structure for conventional wireless sensor networks includes TinyLIME, GSN (Global Sensor Network), ATAG (Abstract Task Graph), Cougar, TinyDB, Mate, Milan, DsWare and SINA. Most of the middleware examples listed before run on top of TinyOS. TinyOS is the de facto operating system for sensor networks that run on motes. Written using the NesC language, TinyOS adopts a component-based model to build sensor network applications in an event-driven operating environment [26]. However, in specific application domains they usually are not so effective in providing efficient and optimized support to specific tasks at communication, sensor and processing levels. [28]

The operating systems, along with its hardware drivers, concurrency mechanisms, and communication channels, is the basis of each middleware.

2.4.1 Operating Systems

Unlike traditional operating systems, operating systems for WESCN must tightly integrate wireless connectivity. The use of operating systems (OS), especially embedded and real-time OS, running on the embedded processors brings many benefits to the development and run-time performance of embedded smart cameras. These include support for memory management, networking, inter-process communication, real-time computing, and high-level design languages such as C and C++. Software development tool compatibility is an issue that should be considered when selecting an operating system for embedded smart camera development.

2.5 State-of-the Art Wireless Embedded Smart Cameras

The goal of this section is to present and describe some industry and research issued wireless embedded smart cameras.

2.5.1 Classification of WESCN

In [1] Akyildis et al. provided a classification based on experimental research, *Commercial of-the-shelf* platforms, such us Stargate/webcam, CMUcam3 and Imote, *Research prototype* such us Mesheye, WiCA, Cyclops and capsule; and *API* examples are WiSNAP and AER.

2.5.2 Examples of Wireless Embedded Smart Cameras

CMUcam3 The hardware platform consists of a color CMOS camera, a frame buffer, a low cost 32-bit ARM7TDMI microcontroller, and an MMC memory card slot. The CMUcam3 also includes 4 servo ports, enabling one to create entire, working robots using the CMUcam3 board as the only requisite robot processor. Custom C code can be developed using an optimized GNU toolchain and executables can be flashed onto the board using a serial port without external downloading hardware. The development platform includes a virtual camera target allowing for rapid application development exclusively on a PC. The software environment comes with numerous open source example applications and libraries including JPEG compression, frame differencing, color tracking, convolutions, histogramming, edge detection, servo control, connected component analysis, FAT file system support, and a face detector.

CITRIC The wireless embedded smart camera platform employed in our experiments is a CITRIC mote [29] . It consists of a camera board and a wireless mote. The camera board is composed of a CMOS image sensor, a fixed-point microprocessor, external memories and other supporting circuits. The camera is capable of operating at 15 frames per second (fps) in VGA and lower resolutions. The microprocessor PXA270 is a fixed-point processor with a maximum speed of 624MHz and 256KB of internal SRAM. Besides the internal memory of the microprocessor, the PXA270 is connected to a 64MB of SDRAM and 16MB of NOR FLASH. An embedded Linux system runs on the camera board. Each camera board connects to a wireless mote via a serial port.

The wireless mote employed is a TelosB mote from Crossbow Technology. The TelosB uses a Texas Instruments MSP430 microcontroller and Chipcon CC2420 IEEE 802.15.4-compliant radio [29]. The maximum data rate of the TelosB is 250kbps.

A more detail analysis for different embedded camera platforms and its main features are presented in [1, 16]

2.6 Applications of WECSN

WESCN have been traditionally used in surveillance and security applications, while more novel applications arise in :

2.6.1 Intelligent Video Surveillance Systems (IVSS)

In video surveillance applications, typical tasks of smart cameras include motion detection, intrusion detection, etc. Large scale of wireless smart camera can extend the ability of already existing approaches monitoring public places. The system can detect and inform about an event, then record the event for previous forensic applications. cited the paper that talked specifically aboout this

2.6.2 Industry Machine Vision

Industrial machine vision is probably the most mature application area for smart cameras, where these cameras perform tasks such as bar code recognition, parts inspection, surface inspection, fault detection, and objects counting and sorting.

2.6.3 Intelligent Transportation Systems

Generally speaking, the application and algorithmic requirements for ITS are quite similar to those of IVSS. These requirements can be quite different for automobile applications, however, where high-speed imaging and processing are often needed, imposing higher level of demand on both hardware and software. Increased robustness is also required for car mounted cameras to deal with varying weather conditions, speeds, road conditions, car vibrations. CMOS image sensors can overcome problems like large intensity contrasts due to weather conditions or road lights and further blooming, which is an inherent weakness of existing CCD image sensors.

2.6.4 Automobile Applications

Intelligent vehicles will form an integral aspect of the next generation technology of ITS. Smart camera-powered intelligent vehicles will have the comprehensive capability of monitoring the vehicle environment including the drivers state and attention inside of the vehicle as well as detecting roads and obstacles outside the vehicle, so as to provide assistance to drivers and avoid accidents in emergencies. However, building and integrating smart cameras into vehicles is not an easy task: On one hand the algorithms require considerable computing power to work reliably in real-time and under a wide range of lighting conditions. On the other hand, the cost must be kept low, the package size must be small and the power consumption must be low cited(). Applications of smart cameras in intelligent vehicles include lane departure detection, cruise control, parking assistance, blind-spot warning, driver fatigue detection, occupant classification and identification, obstacle and pedestrian detection, intersection-collision warning,

overtaking vehicle detection [17].

2.6.5 Personal and Health Care

Ubiquitous personal healthcare systems could help to monitor a patient's status and provide intrinsic and extrinsic information about their daily activities that is needed to interpret the patients' vital data and disease trends. In particular, patients with a chronic condition, that are continuously at risk for a worsening condition, would benefit from technology that can regularly provide useful information. Wireless Embedded Smart Cameras can be use to monitor and study the behavior of elderly people; for instance, falls. The latter information can be used to detect the causes and circumstances under which the event happened. Smart camera networks [30, 31] can infer emergency situations and immediately connect elderly patients with remote assistance services or with relatives.

2.6.6 Gaming

WESCN will find applications in the future prototypes that enhance the effect on the game player. Such as virtual reality games that assimilate touch and sight input, of user as part of the player response [3]

Chapter 3

Energy Consumption and Latency Analysis for Wireless Multimedia Sensor Networks

3.1 Introduction

Energy and bandwidth are limited resources in wireless sensor networks. When wireless vision sensors are used to capture and transfer image and video data, the problems of limited energy and limited bandwidth become even more pronounced, since the amount of data to be handled is much larger compared to scalar sensors [15]. In addition, communication consumes significant energy. Frequent transfer of large-size data requires more power and incurs more communication delay. In many systems, communication is 100 to 1000 times more expensive in energy than computation [32] . Thus, our goal is to reduce the communication cost by decreasing the amount of message traffic. In many applications, the interest is to detect composite and semantically higher-level events based on

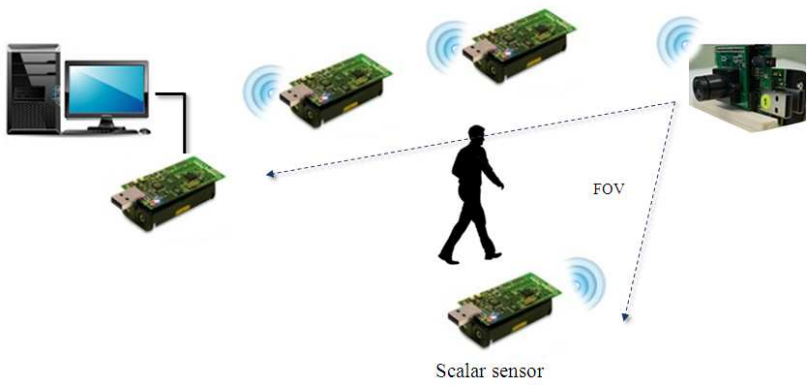


Figure 3.1: Heterogeneous Wireless Multimedia Sensor Network

information from multiple sensors. In existing multimedia sensor network setups [1], each primitive event detected by multimedia nodes are sent to a sink, most probably in a multi-hop manner. Accordingly, the sink or a control center combines information from multiple sensors to make higher-level decisions. In addition, local aggregation can be performed at aggregation points along the path between multiple sensors and the sink. However, event superposition that includes information from spatially distant sensors can only be performed at the sink. In case these composite events are not required to the end user, this creates highly redundant message traffic, consumes a lot of energy, and may overload sink nodes. In addition to the sensor sensing the primitive event, sensors on the multi-hop route also consume energy. Hence, our goal is to push the detection of semantically high-level events within the network, and perform composite event detection in a peer-to-peer (P2P) manner across the heterogeneous and embedded sensors. Accordingly, message traffic, and thus the overall energy consumption of the network will be significantly decreased.

In this chapter, we analyze three different operation scenarios for a heterogeneous sensor network consisting of scalar sensors (for motion detection) and

embedded smart cameras 3.1 . In the first setup, a scalar sensor wakes up the camera mote when it detects motion in the scene. Then, the camera captures a frame, and then, transmits the whole image frame in a multi-hop manner to a sink node. In the second setup, cameras perform local processing and send the images to the sink only if a primitive event is detected. Finally, in the third setup, cameras perform local processing, one camera communicates with another in a P2P manner to detect a composite event, and only when the composite event is detected, they transmit the interesting portion of a frame to the sink. All three operation scenarios are described in detail in Section 3.2. We present a detailed quantitative comparison of these scenarios in terms of the energy consumption when the goal is detecting a composite and semantically high-level event. In addition to providing motivation for and emphasizing the importance of pushing the high-level decision making to the sensor level, this analysis gives quantitative results in terms of savings in energy. We also present a latency analysis for these operation scenarios. The results highlight the need for efficient peer-to-peer communication solutions for wireless smart camera(WMSNs). Using heterogeneous sensors provides energy savings by keeping the low-power scalar sensors active for monitoring, and more power-consuming embedded smart cameras in idle mode until scalar sensors detect an activity. In our testbed, we use CITRIC motes [29] as our embedded smart cameras. A TelosB is attached to the camera boards for wireless communication. The camera board runs with 4 AA batteries, while the TelosB uses 2 AA batteries. We broadcast trigger messages from stand alone TelosBs to emulate the waking up of the cameras by scalar sensors.

3.2 Operation Scenarios for Event Detection

In this section, we describe three different operation scenarios for detecting a composite and high-level event. We consider the event of interest to be a composite event that can be detected by two vision sensors. Accordingly, the composite event is detected if (1) a large vehicle is detected entering a facility through the entrance watched by camera A, and then (2) the same vehicle is detected as parking in a region defined in the view of camera B. It is assumed that cameras A and B have partially overlapping fields of view. In all the operation scenarios, scalar sensors are always active, and camera sensors are idle to save energy. If/when motion is detected, a scalar sensor wakes up nearby camera sensors by broadcasting a trigger message.

3.2.1 Scenario 1: No Local Processing

As mentioned previously, in most existing sensor network setups, individual sensors transfer the captured data to a sink node and/or control center for further processing. To analyze the cost attached to this type of operation, we implement the first scenario, wherein camera sensors do not perform any local processing. After receiving the broadcast image from a scalar sensor, the processor activates the camera board and the sensor warms up. The camera captures a frame and sends the complete image to a sink node (Fig. 3.1) by multi-hop communication. The captured image size is 320×240 and it is transmitted in gray scale format after JPEG compression. This scenario serves as a baseline for the following two scenarios.

In this operation scenario, every time an object enters the facility or every time motion is detected, the scalar sensor will wake up the camera, and the camera will

transmit the whole frame to the sink node. It should be noted that even though the interest is detecting large vehicles, this way of operation will cause an image transfer every time motion is detected, since no local processing is performed.

3.2.2 Scenario 2: Low-level Detection

In this scenario, the embedded camera sensor not only captures images, but also performs local processing. Specifically, it performs foreground detection, and then computes the size of the detected object(s).

As stated above, the event of interest is to detect a large vehicle, which enters a facility through the entrance watched by camera A, and then parks in a region defined on the view of camera B. In this operation scenario, after the camera wakes up, it performs background subtraction to detect the moving object, and then computes its size. If the size of the detected foreground object is larger than a threshold, the camera transmits only the portion of the image containing the object. This way of operation provides savings in two different ways. First, event messages are not transmitted every time motion is detected. Instead, cameras transmit images only if the size of the detected object satisfies a certain criteria. Second, the cameras only transmit the portion of the image containing the object, instead of the whole frame. This scenario serves as the state-of-the-art in WMSNs.

3.2.3 Scenario 3: P2P Composite Event Detection

Cameras perform local processing in this mode as well. If a composite event is defined as a sequence of primitive events across multiple camera views, the first camera in this sequence transmits a message addressed to the next camera when it detects the first primitive event.

The event of interest described above can be defined as a sequence of two primitive events. The first primitive event is detecting the entrance of a large vehicle on the view of camera A. The second primitive event is detecting that vehicle parked in the region specified in the view of camera B. When camera A detects that a large vehicle entered into the scene, it transmits a message addressed to camera B, instead of transmitting a portion of the image to a sink. Compared to the second scenario, P2P composite event detection avoids redundant communication, since the application is not interested in every large vehicle entering the facility. Instead, a higher-level composite event is of interest. If camera B detects the second primitive event, only then an image portion will be transmitted to a sink.

3.3 Experimental Results

In this section, we present the results of a detailed analysis of the energy consumption and latency of the three operation scenarios described above.

3.3.1 Energy Consumption

We measure the energy consumption in each scenario during different parts of the operation including warming up of the camera, processing a frame, and transmitting data. We also measure the energy consumption of the forwarders in multi-hop communication to obtain the overall energy consumption caused by each scenario. For all the results presented below, the communication between a camera sensor and the sink is performed in two hops.

Figure 3.2 shows the overall energy consumption of different operation scenarios. Scenarios 1, 2 and 3 in this figure are the operation scenarios described

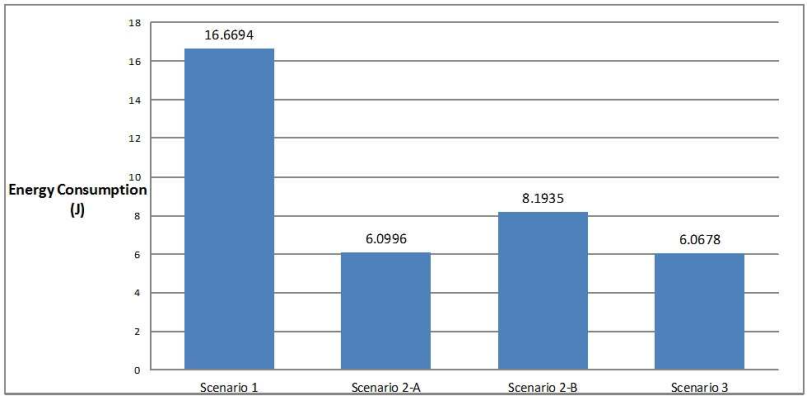


Figure 3.2: The overall energy consumption for different scenarios.

in Sections 3.2.1, 3.2.2 and 3.2.3 , respectively. In Scenario 1, the camera does not perform local processing of the frame, but transmits the whole image frame (320×240) to the sink by two-hop communication. The total energy consumption for this scenario including the energy consumption of the forwarding node is 16.67 J.

Figure 3.3 shows the distribution of the energy consumption among different components. Since no local processing is performed to make decisions, and the whole frame is transferred to the sink, the image data transfer causes the largest

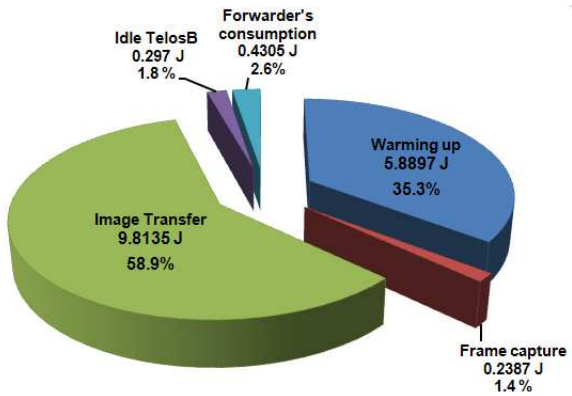


Figure 3.3: **Scenario 1:** The distribution of the consumed energy.

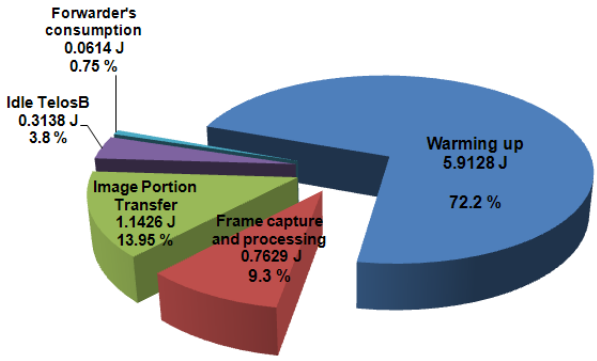


Figure 3.4: **Scenario 2-B:** The distribution of the consumed energy in the second operation scenario when only a portion of the image is transmitted.

energy consumption, i.e., 58.9 %.

In Scenario 2-A, the camera performs local processing to detect foreground objects and to determine their sizes. Since the size of the detected object does not satisfy the specified criteria, the camera does not transmit anything. The overall energy consumption of the camera, including the energy consumption during warming up, frame capturing, foreground detection and size check, is 6.1 J.

In Scenario 2-B, the size of the detected object satisfies the specified criteria, and the camera sends only the portion of the image that contains the object to the sink. The size of this image portion is 50×50. As seen in Fig. 3.2 , the total energy consumption for this scenario including the energy consumption of the forwarding node is 8.19 J, which is significantly less compared to Scenario 1.

Figure 3.4 shows a distribution of the energy consumption among different components. Compared to Fig. 3.3 , this way of operation provides a significant decrease in the energy consumption caused by the image data transfer.

Fig. 3.5 (a) and 3.5 (b) show the operating currents of the camera board during Scenario 1 and Scenario 2-B, respectively.

As can be seen, when local processing is performed, and the interesting por-

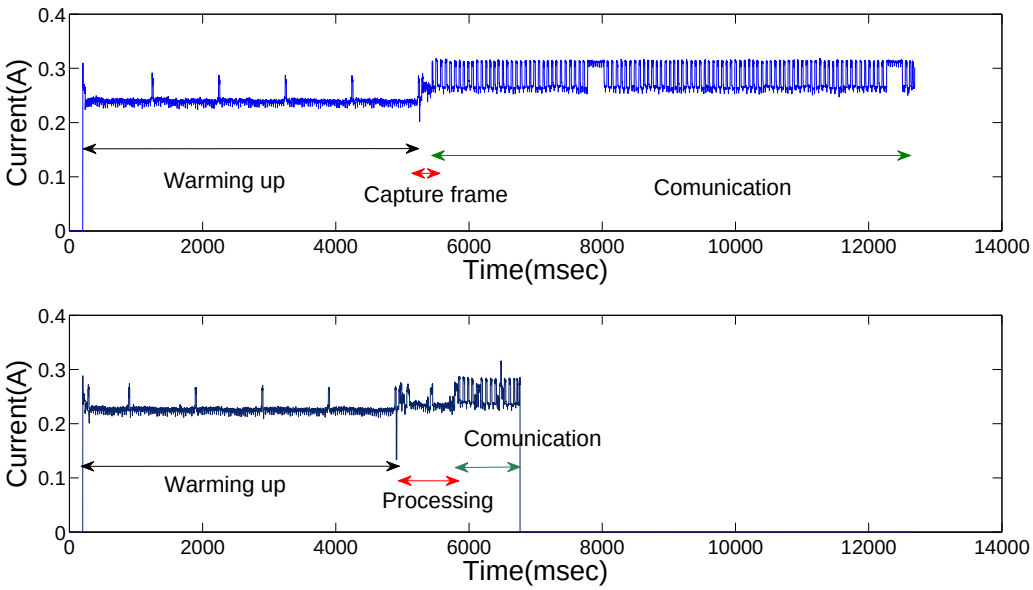


Figure 3.5: Operating current of the camera board when transmitting (a) the whole frame, (b) only the portion of the image containing the detected object.

tion of the image is extracted and transmitted, the amount of latency, and the energy consumption due to image data transfer decrease significantly. In Scenario 3, the camera again performs local processing to detect foreground objects and to determine their sizes. If the size of an object satisfies the specified criteria, the camera sends (in a single hop) a small-size packet to the second camera, which is responsible for detecting the second part of a composite event. This packet contains the label information of the tracked object. Since cameras have partially overlapping fields of view, they can track objects with consistent labels, and can determine if the same object is performing the primitive events in a composite event scenario. The energy consumption of the camera caused by warming up, frame capturing, frame processing and data transfer is 6.07 J.

Figure 3.6 shows the distribution of the energy consumption among different

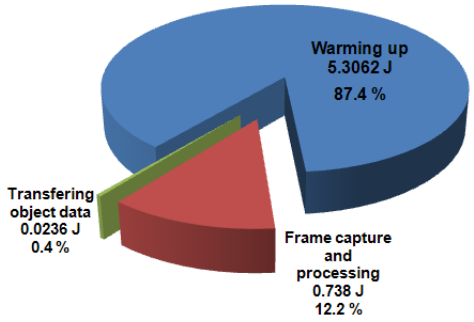


Figure 3.6: **Scenario 3:** The distribution of the consumed energy when the first camera sends information about the object to the second camera.

components.

This analysis provides the motivation for pushing the semantically high-level event detection and decision making to the sensor level by providing quantitative results. It should be noted that the amount of the saved energy becomes much more significant and apparent when we consider the actual composite events that we are interested in. Consider the event of interest described above, where we want to detect large vehicles entering a facility through the entrance watched by camera A, and then parking in a region defined on the view of camera B. Assume that during a day, 10 % of the objects (people, cars, trucks, bikes) entering the facility are large vehicles. Also assume that only 10 % of the large vehicles entering the facility actually park in the restricted region defined on the view of camera B. Let N be the number of objects entering the facility. In Scenario 1, the camera A will wake up, and transmit the complete image to the sink N times. Thus, its energy consumption will be approximately $N \times 16.24$ J. The forwarders energy consumption will be $N \times 0.43$ J.

In Scenario 2, camera A will wake up N times, but $\frac{N \times 9}{10}$ many times it will not transmit anything, since the size of the object will not be large enough (as-

suming the object detection and size check does not fail). It will transmit only the portion of the image containing the object $\frac{N}{10}$ times. The energy consumption of camera A in this case will be approximately $\frac{6.1 \times N \times 9}{10} + \frac{8.1 \times N}{10} \simeq 6.3 \times N$. In scenario 3, camera A will wake up N times, and will send a message packet to camera B, $\frac{N}{10}$ many times. Thus, the energy consumption of camera A will be $\frac{6.1 \times N \times 9}{10} + \frac{6.06 \times N}{10} \simeq 6.09 \times N$. Camera B will transmit an image only $\frac{N}{10}$ times.

Thus compared to Scenario 1, Scenario 2 and Scenario 3 provide 61.21 % and 62.5 % savings, respectively, in the energy consumption of camera A. In addition, Scenario 1 involves an image transfer N times. In Scenario 2, an image portion is transferred $\frac{N}{10}$ times, and in scenario 3 an image portion is transferred only $\frac{N}{100}$ times.

3.3.2 Latency

We also measured latency introduced during these different operation scenarios. It should be noted that in all latency measurements, the measured time intervals include the warming up time of the camera sensor, which is around 5 sec. For the first scenario described in Section 3.2.1, we measured the time interval from camera waking up to a sink node receiving the complete image. The distance between the camera sensor and the sink node is 12 meters, and communication is performed in two hops. The file size for the whole image is 9.5 kB. After camera wakes up, it captures a frame, compresses it, and sends the whole frame to the sink node. We performed this experiment five times, and took the average of all measurements. The average time obtained is 11.85 sec.

For the second scenario described in Section 3.2.2 , we measured two different

latencies. In the first case, camera wakes up, performs foreground object detection, and determines the size of the detected object. If the size of the object is not large enough, the camera does not transmit anything. We measured the time interval between camera waking up and determining if the size of the object satisfies the criteria. This was repeated five times. The average measured time interval is 5.36 sec. If the camera determines that the size of the detected object satisfies the specified criteria, then it transmits only the portion of the frame containing the object. For this case, we measured the time interval starting from camera waking up, including processing of the frame, and ending when the sink receives the transmitted portion of the image completely. Again, the distance between the camera and the sink is 12 meters, and communication is performed in two hops. The size of the image portion that is transmitted is 50×50, and the file size is 1.8 kB. The average measured latency for this case is 6.24 sec.

In the third scenario described in Section 3.2.3 , camera A performs foreground object detection, and determines the size of the detected object. If detected object is large enough, camera A sends a message addressed to camera B containing the label of the detected object. The average measured time from camera A waking up to camera B receiving the message packet is 5.51 sec. Camera A and camera B communicate in single-hop and the average measured latency for communication between them is 0.33 sec.

Figure 3.7 shows the amount of time it takes to complete warming up, processing and communication for all three scenarios.

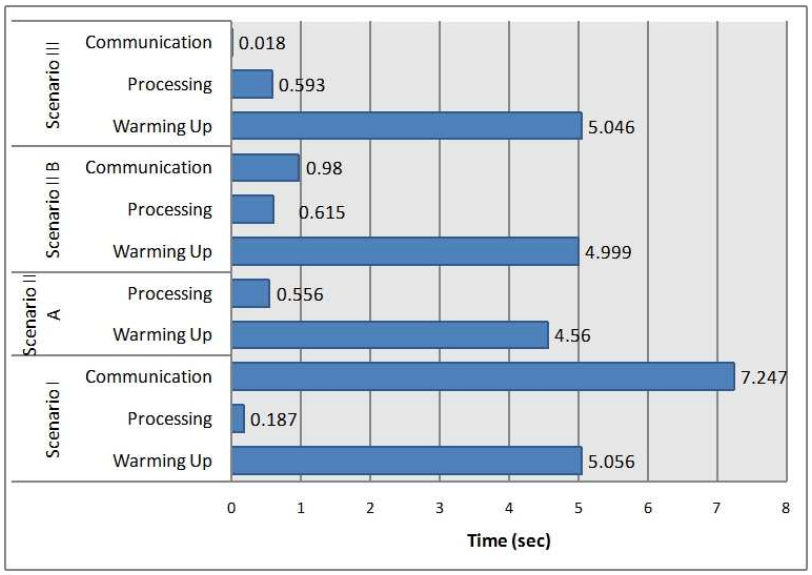


Figure 3.7: The latencies of different components of operation for different scenarios.

Chapter 4

Analysis of the Accuracy-Latency-Energy Tradeoff for Wireless Embedded Camera Networks

4.1 Introduction

Wireless embedded smart cameras are stand-alone units that combine sensing, processing and communication on a single embedded platform [1] . Wireless embedded camera networks have promising applications in surveillance, traffic analysis and wildlife monitoring. Unlike wired camera systems, these cameras have very limited energy, processing power and memory.

Energy consumption and latency are two major concerns in wireless embedded camera networks. Methods and systems have been presented in literature to reduce energy consumption of the cameras, either for image transmission [33, 34],

or object detection and tracking [35, 36, 37]. For object detection and tracking, multi-tier camera sensor networks have been introduced to reduce energy consumption. Kulkarni et al. [35] introduced *SensEye*, which is a two-tier camera sensor network, compared the sensing reliability as well as the energy usage with one-tier systems. Another two-tier system is presented in [36] , which employs cameras with different resolutions. Low resolution cameras continuously determine position, range, and size of moving objects and trigger high resolution cameras. High resolution cameras perform the subsequent image processing. This two-level structure is also used in [37] , wherein a probabilistic algorithm is employed to reduce the sensing work of the lower level cameras. The aforementioned work focuses on two-tier structures and cameras with different resolutions. Overlapping and non-overlapping camera setups are not addressed, and energy-accuracy-latency tradeoffs are not analyzed.

Margi et al. [38] analyzed the energy consumption of each basic task in the camera motes, such as processing, flash memory access, image acquisition and communication. Ko et al. [39] empirically study a camera sensor node which uses Scale Invariant Feature Transform (SIFT) to identify objects in the environment. They analyze the performance (classification accuracy, latency and energy consumption) of SIFT for visual classification on a Blackfin DSP processor. The simulation results are provided. In this chapter, we implement a multi-camera tracking algorithm on actual embedded smart cameras. We provide a detailed quantitative analysis of the accuracy-latency-energy tradeoff for overlapping and non-overlapping camera setups when different-sized data packets are transferred in a wireless manner.

A quick review of the embedded platform use is present in section 4.2 and a more details of the embedded camera platform are described in Section 2.5. All

the processing, including foreground detection and tracking, is performed on the microprocessor of the camera boards.

The rest of the chapter is organized as follows: Section 4.2 presents the embedded smart camera platform employed in our experiments. Section 4.3 gives a detailed description of the camera configurations and scenarios considered to analyze the energy consumption, latency and accuracy of the system. Section 4.4 briefly discusses the communication protocols in different configurations and the experimental results are provided in Section 4.5.

4.2 The Embedded Smart Camera Platform

The wireless embedded smart camera platform employed in our experiments is a CITRIC mote. It consists of a camera board and a wireless mote. The camera board is composed of a CMOS image sensor, a microprocessor, external memories and other supporting circuits. The image array is capable of operating at up to 30 fps in VGA and lower resolutions. The microprocessor PXA270 is a fixed-point processor with a maximum speed of 624MHz and 256KB of internal SRAM. An embedded Linux system runs on the camera board. The wireless mote employed is a TelosB mote. The TelosB uses an IEEE 802.15.4-compliant radio. The maximum data rate of the TelosB is 250kbps.

To measure the energy consumption of the platform, a National Instruments DAQ device is used. The voltage as well as the current of each mote is measured. Since totally 8 channels are used in the DAQ device (4 voltages and 4 currents), and the maximum total sampling rate of the device is 250K samples/s, the sampling rate of each parameter is 31250 samples/s. For the purpose of measuring communication delay, all the motes are connected to a PC using USB cables to

record the communications among the motes and the time stamp of each packet.

4.3 Analysis of Energy Consumption, Latency and Accuracy

We have used two different camera configurations (partially overlapping and non-overlapping) and performed object tracking for different scenarios. Within these scenarios, different amount of data is exchanged to perform a detailed quantitative analysis of the accuracy-latency-energy consumption tradeoff. To this end, the energy consumption, accuracy, and latency are measured when tracking one or two objects with very similar or different colors. In the following, the camera configurations and the deployed scenarios are described.

4.3.1 Camera Configurations

Two different camera configurations, i.e., partially overlapping and non-overlapping, are used for the experiments as described below.

4.3.1.1 Partially Overlapping Cameras

For this setup, we installed four embedded smart cameras (CITRIC motes) with partially overlapping fields of view as seen in Fig. 4.1. Since camera views are overlapping, they only exchange the location information of objects, specifically the x and y coordinates of the midpoint of the bottom line of the bounding boxes around the objects (the red cross shown in Fig. 4.1).

Consistent labeling of objects is performed as follows:

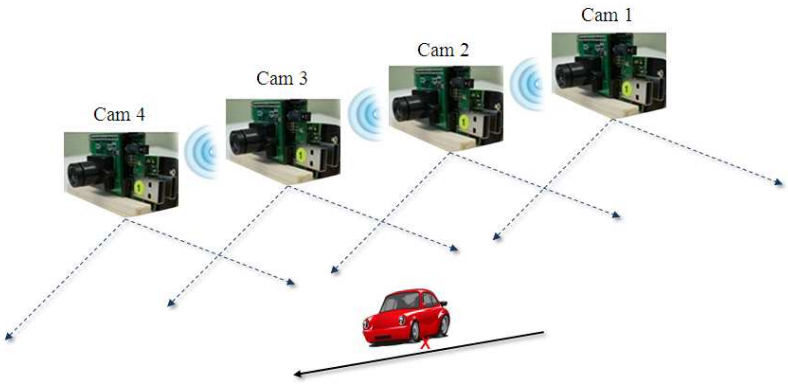


Figure 4.1: Camera configuration with four partially overlapping cameras.

i: The homography matrices are estimated between camera pairs. When a new objects enters its view, a camera uses the field of view (FOV) lines to determine which camera(s) can also see this object. We recover the FOV lines off-line as described in [40].

ii: The camera transmits a message addressed to the camera(s) that has already been tracking the object. This message contains the objects x and y coordinates and its temporary label.

iii: The receiving camera uses the homography matrix to convert the received point to its own coordinate system, and finds the object in its view that is closest to this point. It sends a reply packet to the requesting camera, and this packet includes the answer label and the received temporary label.

Figure 4.2 shows representative frames from an experiment with four overlapping cameras. The remote-controlled car enters the scene in the view of Cam1, and gets a new label 10. At time instant $t = t1$, it enters the view of Cam2, and Cam2 gives it a temporary label 0. Then Cam2 gets the correct label 10 from Cam1 via location exchange. At time instant $t = t2$, it enters the view of Cam3,

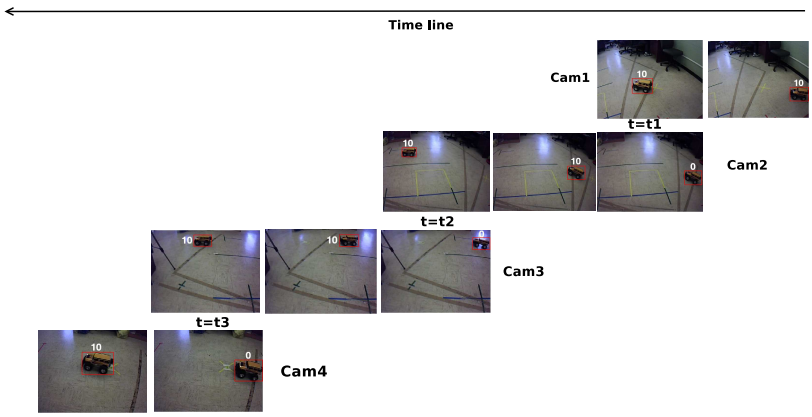


Figure 4.2: Representative frames from the setup with four overlapping camera views.

and gets a temporary label 0. Then, Cam3 receives the correct label 10 from Cam2 via location exchange. Same steps happen after time instant $t = t3$.

4.3.1.2 Non-overlapping Cameras

In the second configuration, the camera views do not overlap, i.e. there is a spatial discontinuity between cameras as shown in Fig. 4.3 . In this case, cameras cannot perform consistent labeling by location exchange. Instead, larger amount of data needs to be transmitted to consistently track objects across different camera views. Even if the cameras are initially installed with overlapping views, potential camera failures can cause non overlapping camera setups and blind regions. Thus, analyzing this setup is important.

4.3.2 Tracking Scenarios

In the following, we describe several tracking scenarios that differ in terms of the type and amount of data transferred between cameras.

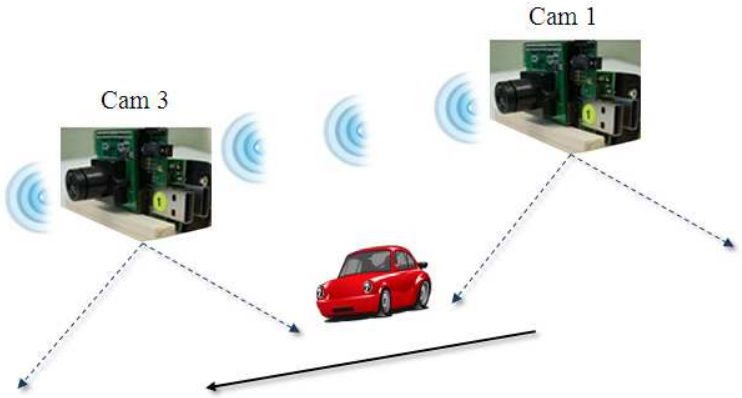


Figure 4.3: Camera configuration with two non-overlapping cameras.

4.3.2.1 Scenario I: Gray-level Histogram Exchange

In this scenario, gray-level histograms are transferred to match objects across non-overlapping cameras. For the histograms, we use 32 bins and the Y channel of the YUV color space. For a pixel, the value of the Y channel ranges between 0 and 255. The background subtraction and tracking algorithms running on the camera boards detect moving objects, build their histograms and track them on a cameras view. In order to detect the salient moving objects, we use the algorithm we presented in [41] . The details of the tracking algorithm can be found in [42] .

In Fig. 4.3 , after an objects leaves the view of Camera 1, this camera saves the objects histogram in memory. When a new objects enters into the view of Camera 3, this camera gives a temporary label 0 to this object, and transmits the objects gray-level histogram. The receiving camera (Camera 1) calculates the Bhattacharyya coefficient [43] between the received histogram and its saved histograms to find the best match. When the match score is larger than a predefined threshold, Camera 1 sends a reply packet that includes the label of the matched

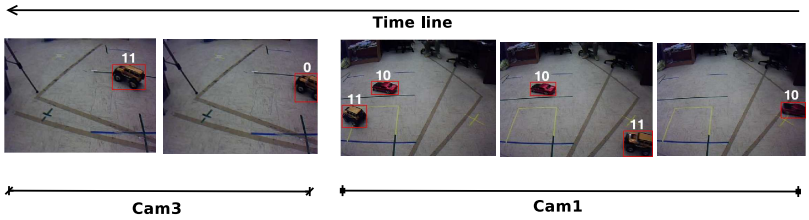


Figure 4.4: Representative frames from a two non-overlapping camera setup.

object. Camera 3 then assigns the received label to the object with the temporary label. The Bhattacharyya coefficient is derived from the sample data by using:

$$\hat{\rho} \equiv \rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] = \sum_{u=1}^{m} \sqrt{\hat{P}_u(\mathbf{y})\hat{q}_u}, \tag{4.1}$$

where $\hat{\mathbf{q}} = \{\hat{q}_u\}_{u=1\ldots m}$ and $\hat{\mathbf{p}}(\mathbf{y}) = \{\hat{p}_u(\mathbf{y})\}_{u=1\ldots m}$ are the probabilities estimated from the m-bin histogram of the model in the tracker and the candidate blobs, respectively.

In our system, cameras can save 10 object histograms in their memory. Under Scenario I, we analyzed the energy consumption and latency when tracking one and two objects. We also analyzed the accuracy when tracking two objects with similar brightness levels and two objects with different brightness levels. Results are discussed in Section 4.5.

Figure 4.4 shows representative frames from an experiment with two non-overlapping cameras. Two remote-controlled cars are tracked in the view of Cam 1 first. Then, the car with label 11 leaves the view of Cam1, and after going through the blind region in between, it enters the view of Cam3. It first gets a temporary label 0. Cam3 sends the cars gray-level histogram, and Cam1 sends the correct label 11 back.

4.3.2.2 Scenario II: Color Histogram Exchange

In this scenario, each camera builds color histograms of the moving objects. Information from additional channels helps especially when tracking objects with similar brightness levels. For each channel (Y, U and V) the range of values is divided into 32 bins. This creates a 32^3 dimensional array, and this generated histogram is usually sparse, which means that there are not too many nonzero values in the array. To decrease the memory requirement, the amount of the transmitted data and the energy consumption of the embedded smart cameras, we compress the color histogram before saving it and/or transmitting it over wireless channel. Only the nonzero values and the indices of the nonzero values in this array are transmitted.

We analyzed the energy consumption, accuracy and latency when tracking one or two objects, and when tracking objects with very similar or with different colors. As will be discussed in Section 4.5, there is a tradeoff between the amount of data transmitted and the energy consumption as well as the latency introduced. Also, when the amount of transmitted data increases, which means that richer image descriptors are transmitted across the cameras, the accuracy of consistent tracking also increases. However, the energy consumption also increases significantly.

4.4 Communication Protocol

In the overlapping camera setup, when a new object enters into a cameras view, this camera transmits a message addressed to the camera(s) that has already been tracking the object. This message contains the objects x and y coordinates and its temporary label. The receiving camera sends a reply packet to the requesting

camera, and this packet includes the answer label and the received temporary label. The payload of the request and reply packets are 4 and 2 bytes, respectively. In Scenario I of the non-overlapping configuration, gray-level histograms are sent. The message packet contains 32-byte histogram information and the temporary label of the object. Thus, the total payload is 33 bytes. In Scenario II of the non-overlapping configuration, compressed color histograms are transmitted and the size of the packet varies according to the number of nonzero entries in the histogram. Thus, it depends on the detected object. The TelosB allows sending 114 bytes of payload. Thus, the compressed histogram is divided into multiple packets. Moreover, only char type data can be sent through this packet structure. However, the indices of the nonzero entries in the 32^3 dimensional array can have large values. Thus, the index information is divided into three different bytes. As a result, for each nonzero value in the array, 4 bytes of information is sent resulting in a total packet size of $4 \times N$ bytes, where N is the number of nonzero entries in the histogram.

4.5 Experimental Results

In our experiments, we started with a 4-camera setup shown in Fig. 4.1, where cameras have partially overlapping fields of view. Objects enter the scene through the first cameras view, and are tracked across Cameras 1 through 4. We then studied the non-overlapping setup shown in Fig. 4.3 assuming that Cameras 2 and 4 have failed. Objects are tracked across Camera 1 and 3. We analyzed the energy consumption, latency and accuracy of the system when tracking objects across different camera views. We repeated every experiment for every scenario 10 times and present the average value of the obtained results. This study provides a

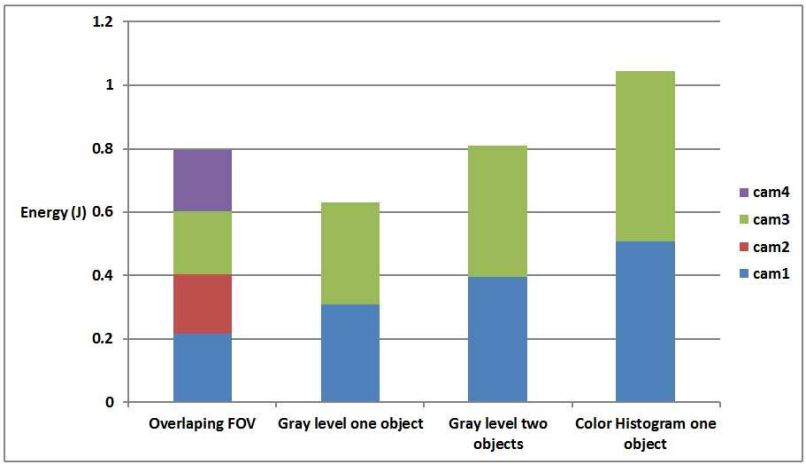


Figure 4.5: Energy consumption for the 4-camera overlapping and 2-camera non-overlapping setups for different types of data transfer.

quantitative analysis of the accuracy-latency-energy tradeoff.

4.5.1 Energy Consumption

In Figure 4.5, the resulting energy consumption is shown for each camera during a time window that starts when a target object enters the cameras view and ends when the object leaves the cameras view. This amount includes the energy consumed during message exchanges. We also looked at cases when there were one or two objects in the scene.

In Figure 4.5, the total energy consumption for all the cameras in the system can also be seen for overlapping and non-overlapping setups. The results show that deploying four overlapping cameras and communicating less data provides 23.56 % savings in energy consumption compared to using two non-overlapping cameras and exchanging compressed color histograms. The total energy consumption of Scenario I when tracking one object is 20.75 % less than the 4-camera setup. However, as will be discussed in Section 4.5.4, it results in 15.8 % lower

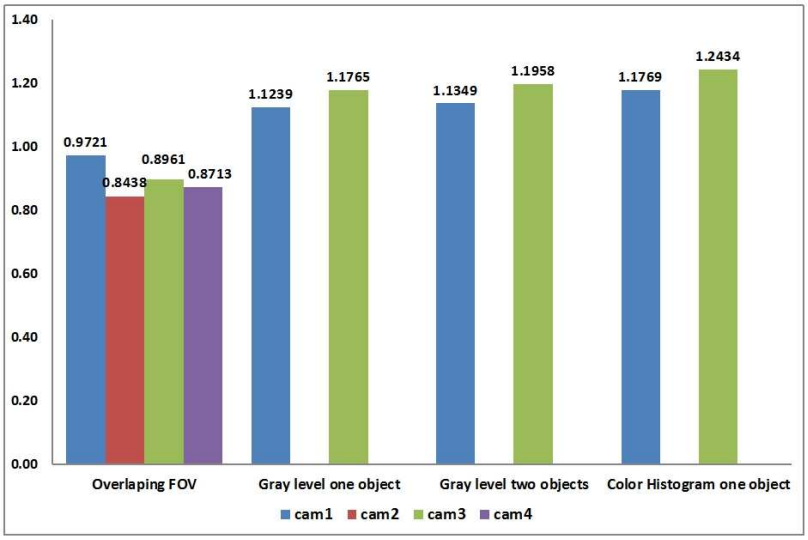


Figure 4.6: Average power per camera.

accuracy/reliability.

4.5.2 Average Power

In addition to energy consumption, we measured the average power consumption of the cameras for the camera configurations and scenarios described in Section 4.5.1. Figure 4.6 7 shows the obtained values.

4.5.3 Latency

We measured the latency from the time a camera detects a new object until the time it receives the answer label from another camera and assigns the received label to the newly detected object. Figure 4.7 shows the measured latency values for different scenarios. As can be seen, sending larger-sized packets introduces longer delays.

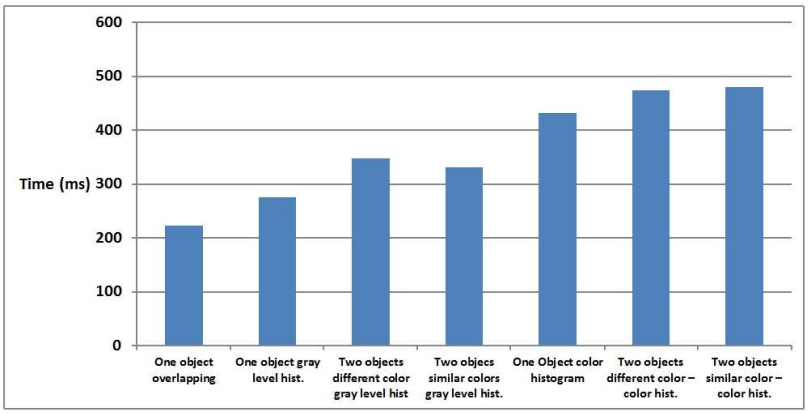


Figure 4.7: Latency for different setups and sized packets .

4.5.4 Reliability

We measured the accuracy of tracking objects consistently across different cameras for overlapping and non-overlapping configurations. The results are displayed in Fig. 4.8. As seen in this figure, when using cameras with overlapping fields of view, the reliability is very high for uncrowded scenes. Since only location data is exchanged, the reliability will decrease for densely crowded scenes. Also, the reliability of using only gray-level histogram for non-overlapping cameras decreases especially when there are multiple objects in the scene with similar brightness values. Using all three channels increases the accuracy with the cost of higher delays and higher energy consumption.

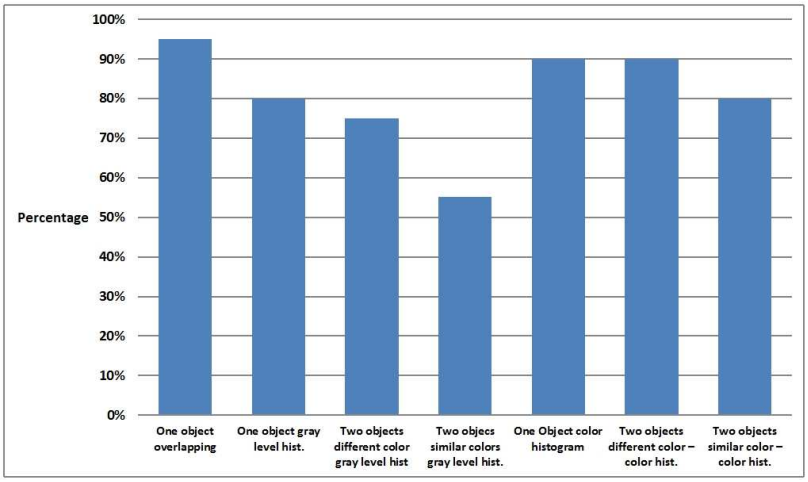


Figure 4.8: Accuracy of correctly labeling an object across different camera views for the 4-camera overlapping and 2-camera non-overlapping setups.

Chapter 5

Fall Detection for Eldercare: Motivation and Preliminary work

5.1 Introduction

In this chapter, we present the motivation for, and describe the design and implementation of a wireless embedded smart camera application, namely fall detection for eldercare. This approach is part of a novel suite of applications being developed to address healthcare-related problems.

Ubiquitous personal healthcare systems could help to monitor a patient's status (vital functions) and provide intrinsic and extrinsic information about their daily activities that is needed to interpret the patients' vital data and disease trends. In particular, patients with a chronic condition, that are continuously at risk for a worsening condition, would benefit from technology that can regularly provide coaching support and objective feedback to healthcare providers. By using such personal monitoring solutions, health risk could be reduced and the patient's comfort increased.

The fact that western societies are aging rapidly in the last decades, has resulted in the necessity of systems that will be able to monitor vulnerable persons, always respecting their privacy. Among the elders, falls and their consequences are among the very major problems. Age UK, which offers products and services that are designed to help the Aged, states that up to one in three people aged 65 and over fall each year [44]. The costs of managing falls by the National Health Service (UK) have increased from £1.5 billion in 2003 [45] to £1.7 billion in 2010 [44]. Detecting falls to get immediate help reduces the risk of hospitalization by 26% and death by more than 80% [21]. The U.S. Department of Health & Human Services reports that in 2000, the total direct cost of all fall injuries for people 65 and older exceeded $19 billion. The financial toll for older adult falls is expected to increase as the population ages, and may reach $54.9 billion by 2020 [46] . Thus, different studies [30, 47, 48] highlight a evidences regarding falls that may be useful to researchers in the field.

Networks of wearable sensors, such as accelerometers, gyroscopes, EKG, pulse oximeters[49, 50, 51], and camera [30, 31] sensors can infer emergency situations and immediately connect elderly patients with remote assistance services or relatives. Telemedicine sensor networks can be integrated with third generation multimedia networks to provided ubiquitous healthcare services. Patients will carry medical sensors to monitor parameters such us body temperature, blood pressure, pulse oximetry, EKG, and breathing activity. Moreover, remote medical centers will perform advance remote monitoring of their patients via audio and video sensors, location and activity sensors, which can also be embedded in wrist devices or the so-called smart " Band-Aids ". [1].

In order to monitor and gather useful information, a wireless smart camera-based approach is proposed. A wireless embedded smart camera can be used to

monitor and study the behavior of elderly people, and detect falls. A camera can also observe the circumstances under which the event happened.

5.2 Related Work

A lot of effort has been invested in developing state-of-the-art technology for detecting falls of elder patients. Detail information of this topic is presented in [52]. More successful approaches use either accelerometers, gyroscopes, RFID sensors and vision systems or a combination of them. [30, 31, 53, 54, 27, 55, 56]. However, they are seldomly evaluated with real users and real-world deployment [57, 58, 59].

There have not been in-depth clinical studies of the feasibility of wireless vision monitoring systems for elderly patients, yet there are a few notable exceptions. The most notable approach is the University of Missouri and its program called "TigerPlace" [60]. They used calibrated features acquired from background subtraction results (silhouettes) of multiple calibrated cameras, along with the 3D voxel object formed from the intersection of those multiple silhouettes in a volume space [61].

Diraco et al. [30] present a method for fall detection in 3D range image that combines information about the 3D position of the centroid of the people with the detection of inactivity. This approach starts with a calibration procedure which searches for different planes in the scene selecting the one that accomplishes the floor plane constraints. Subsequently, the moving regions are detected in real-time by applying a Bayesian segmentation to the whole 3D points cloud. The distance of the 3D human centroid from the floor plane is evaluated by using the previously defined calibration parameters and the corresponding trend is used as

feature in a thresholding-based clustering for fall detection.

Grassi et al. [54] propose a multi-sensor system, where two kinds of devices are used: a MEMS wearable wireless accelerometer with on-board fall detection algorithms and a 3D Time-of-Flight camera. An embedded computing system receives the possible fall alarm data from the two sub-sensory systems and their associated level of confidence. Texeira et al. [55] also utilize a multi-sensor system(accelerometer + static camera). They use a distance measure between signals comprised of timestamps of gait landmarks, and utilize it to identify each tracked person from the video by pairing them with a wearable accelerometer node. A detailed review about characteristics of fall and vision-based approaches can be found in [62].

Few approaches have been propose related to embedded cameras. Culurciello et al. [63] present an address-event vision system. They use an asynchronous temporal contrast vision sensor which features sub millisecond temporal resolution. A lightweight algorithm computes an instantaneous motion vector and reports fall events. They claim to protect the patient's privacy since the address event imager takes no image snapshot. Williams et. al. [64], proposed a distributed network of smart cameras using Cyclops cameras running on the Crossbow MICAz platform which uses a decentralized procedure for computing inter-image homographies that allows the location of a fall to be reported in 2D world coordinates by calibrating only one camera. Other methods [31, 65] use distributed algorithms in embedded smart cameras to recognized human posture; for instance, falls.

Among the literature we can distinguish two main approaches. They use either embedded or centralized systems to process the image data and extract useful information to detect falls. All the methods described above are static camera-based and people are being watched. To the best of our knowledge no previous

work has been reported to detect falls using wearable cameras, which represent a novelty of our approach.

5.3 Our Approach

Cameras can provided useful information about surroundings, and also possible cause of falls. However one major issue that needs to be addresses when using cameras is the privacy. Elder people want to preserve their privacy during their daily activities. Each individual should be able to have control of information about him- or her-self [66]. A detailed analysis of application of smart cameras in assisting living is presented in [21], where the problem of privacy in this field is explain in detail. In this chapter, we present a method to detect falls using wearable wireless embedded smart cameras. The proposed method is suitable to be deployed in ubiquitous healthcare, since it does not require camera calibration, training or previous information about the environment. In our approach, the privacy issue is addressed in four different ways: (i) the data is transmitted only when a fall occurs, (ii) in the event of a fall, the images could be saved locally, and only a message could be transferred; (iii) the images are of the surroundings of people and not of the people themselves, since the cameras are worn by them. In other words, they are not *being watched*; (iv) people can.

An advantage of using cameras over approaches using accelerometers and gyroscopes [67, 68, 56] is that images can be saved for later analysis. On the other hand, static distributed smart cameras just cover limited field-of-view(s). Wearable cameras move with people and cover unlimited areas, including places, such as attics, basements, back-front yards or even public places where usually traditional systems for fall detection are not usually located and have conditions

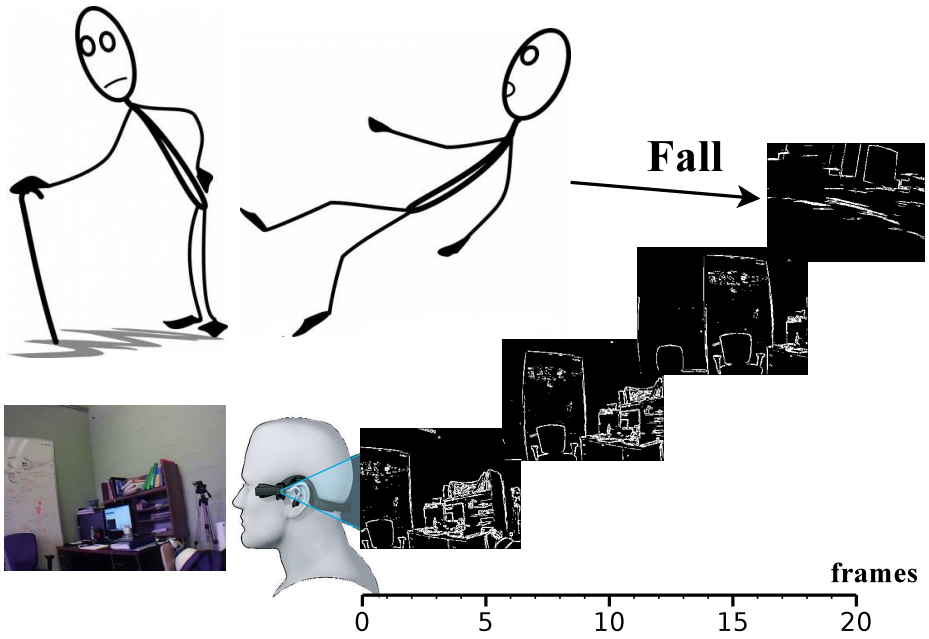


Figure 5.1: Overview

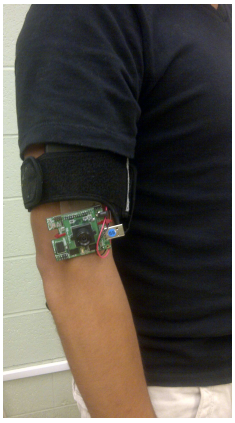
that increase the probability of falling.

In this chapter, we propose a novel approach to detect falls of elderly people by using wearable embedded smart cameras. There are many challenges that need to be addressed. One major challenge is that the camera is mobile, and the background changes continuously. Also, the processing power and memory are limited resources.

Our proposed approach uses wireless embedded smart cameras which are constrained in terms of battery-power, processing capability, memory and achievable data rate [1]. The main objective is to capture the surroundings of the user, and then analyze it to determine abrupt changes, such as falls. Figure 5.1 shows an overview of our system.

With rapid improvement and miniaturization in hardware, a single embedded device can be equipped with an image sensor, processing unit and communication

(a) Wearable Cameras



(b) CITRIC Camera use as a prototype

Figure 5.2: Examples of wearable cameras

unit. Some of the actual devices that have embedded cameras are shown in figure 5.2(a) and the CITRIC platform [29] used to test our algorithm is shown in figure 5.2(b).

5.4 Preliminary Work: Comparison of different approaches for change detection

In this chapter we develop a fall detection system using image features, such as edges. The final goal is to implement the algorithm into a wireless embedded camera. In order to develop a fall detection algorithm, we used Matlab because it provides tools for a rapid development and implementation of vision algorithms. We use three different approaches: Hough Transform, Optical Flow, and Cross-correlation. All the previous methods have a common first step which is Edge Detection. Edge detection is a mature technique and is relatively easy to carry out. We decided to implement a Sobel filter to obtain the edges directly into

a embedded camera(CITRIC platform), the results are saved and then feed into Matlab to implement the algorithms mentioned above

5.4.1 Edge Detection

On the embedded camera (CITRIC platform), edge detection is performed using Sobel filter. Sobel Operator is a differential operator computing an approximation of the gradient of the image intensity. It is very fast to apply since only a small window (3×3 kernel) is convolved with the whole image. The thresholded square root of th absolute values in x and y derivatives are used as features [69, 70].

$$\mathbf{S}_{edge}(x,y) = (\sqrt{\mathbf{G}_x + \mathbf{G}_x}) \tag{5.1}$$

where:

$$\mathbf{G}_x(x,y) = \frac{\partial I}{\partial x} \approx sobel_x * I, \mathbf{G}_y(x,y) = \frac{\partial I}{\partial y} \approx sobel_y * I \tag{5.2}$$

$$\mathbf{G}_x(x,y) = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{I} \quad \text{and} \quad \mathbf{G}_y(x,y) = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{I} \tag{5.3}$$

$\mathbf{G}(x,y)$ is the sobel operator and denote convolution with the x and y components of the Sobel Kernel, one for horizontal changes, and one for vertical.

In other words, $\mathbf{S}_{edge}(x,y)$ calculates the gradient magnitude of the image intensity at each point and it is an image which contains the abrupt changes from dark to light (edges). $\mathbf{G}_x$ and $\mathbf{G}_y$ are two images which at each point contain the

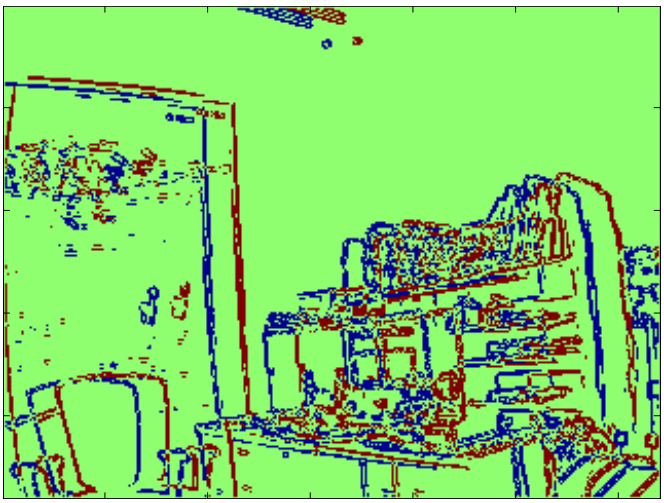


Figure 5.3: Temporal Differencing

horizontal and vertical derivative approximations.

The algorithm presented above was implemented in an actual embedded camera, then the images which contain the edges are saved and transmitted to the PC where we process them using Matlab.

5.4.2 Hough Transform

The first step towards apply a hough transform is to obtain the temporal differencing between the actual and previous image. Figure 5.3 shows the temporal differencing between two consecutive frames. The approach that we used to detect abrupt changes in the scene uses Hough transform to detect lines with high scores and then track them in a spatial and temporal manner. However, the line detection is not temporal coherent. In other words, the detection of the same edge-line and the probability of matching them in the next frame is very low due to variations in brightness. Figure 5.4 shows the lines obtained using Hough transform; lines in red and green represent the actual and previous frame respectively.

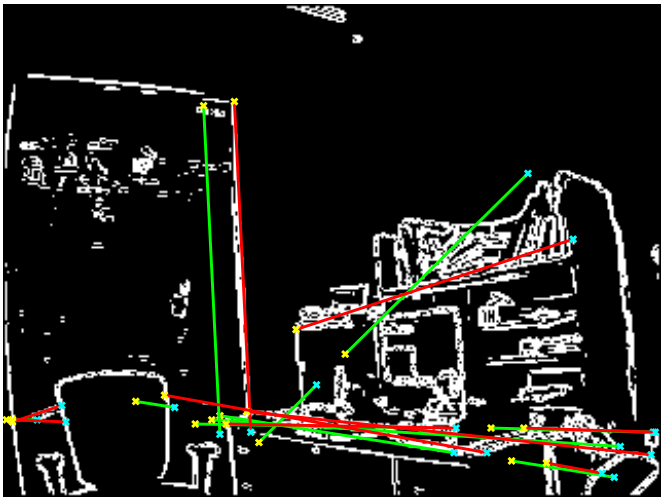


Figure 5.4: Hough Transform. Lines previous and actual frame

5.4.3 Optical Flow

One way of detecting motion is by using optical flow. Motion detection works on the basis of frame differencing. This can be described as the simple subtraction of images acquired at different instants in time which makes motion detection possible.

Figure 5.5 presents an average of the magnitude and direction of the vectors obtained with the optical flow (Lucas-Kanade Method). This approach gives a sense of the direction and the velocity of the estimated motion. In a normal situation (not falling) consecutive frames keeps approximately the same direction and the magnitude of the velocity has small variations.

However, optical flow changes dramatically in highly textured regions, around moving boundaries, at depth discontinuities, etc. Resulting errors propagate across the entire optical flow solution.

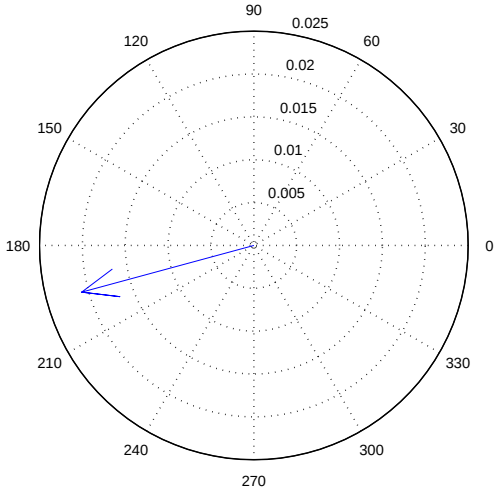


Figure 5.5: Average Optical Flow

5.4.4 Cross-correlation

Cross-correlation provides a good reference as to whether scene has changed by using two consecutive frames. However this approach is computationally too expensive for real-time applications using resource-constrained devices. The cross correlation between these two frames is computed and the result is shown in figure 5.6.

5.4.4.1 Maximum Normalized Cross-correlation Ratio

In order to reduce the dependencies on empirical thresholds, we calculate the normalized cross-correlation and display it as a surface plot. The peak of the cross-correlation matrix occurs where the two consecutive frames are best correlated. The normalized cross correlation plot shows that when the value exceeds the set threshold, the target is identified. The literature in this field is extensive; especially, in medical image analysis. The researchers in this field concluded that a good match occurs when the maximum peak is above 0.7.

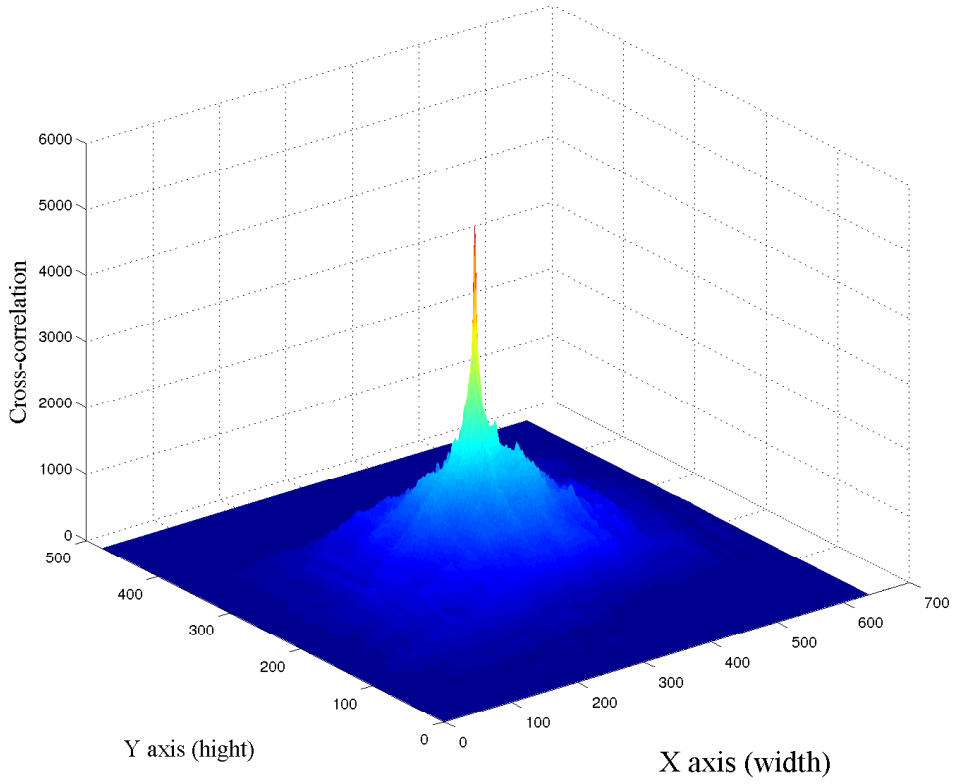


Figure 5.6: Cross correlation between actual and previous frame

We use normalized cross correlation for detecting targeted events (falls) in a frame sequence. We implement the algorithm as follows:

- First we reduce the size of the image being test in order to ensure that the cross correlation is computed over a lesser area thereby saving computation time. During this process the borders of the image are continuously changing if the camera is moving. Thus, it is enough to compare the reduced centered image in this case (300×220)
- After this, we detect the target event based on the ratio between the global maximum correlation value and the surrounding local peaks.

The algorithm localizes the coordinates of the global maxima, then takes a grid of 3×3 surrounded the global maxima and looks for the local maximas at each grid's location. The ratio between the global and average local maxima is taken as a normalized ratio which highlight the targeted event (fall). Figure 5.7 shows the normalized ratio from a scenario where a person walks normally and then falls, simulating an emergency. Additionally, we consider another approach which can help to determine a targeted event. In this approach we plot the magnitude of the ratio vs x and y coordinates, as is shown in figure 5.8. We clearly distinguish that normalized ratios at each frame, which are similar to the previous, posses a global maxima which falls in a centered region and has similar amplitude. In the other hand consecutive frames which are not similar and could represent a fall are out of this centered region. Also, the amplitude is smaller than frames with hight similarity ratio.

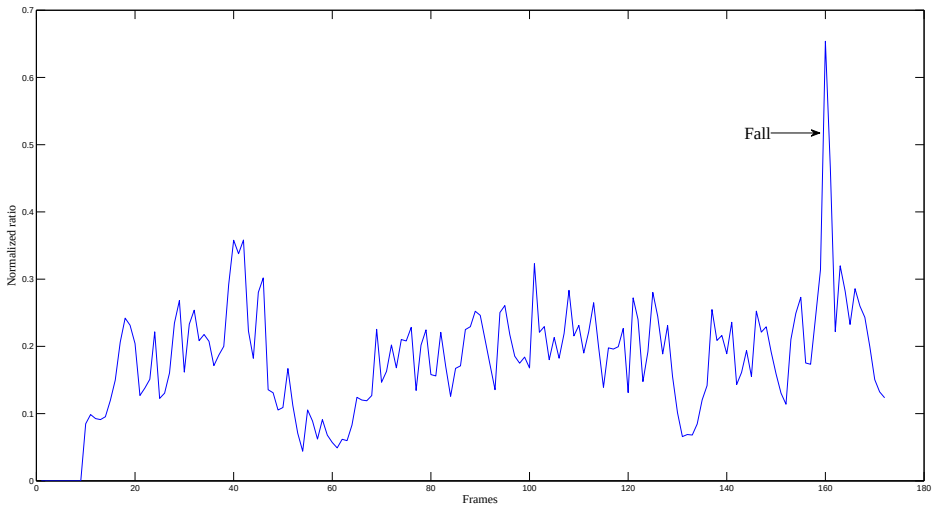


Figure 5.7: Normalized ratio maximum cross-correlation peak

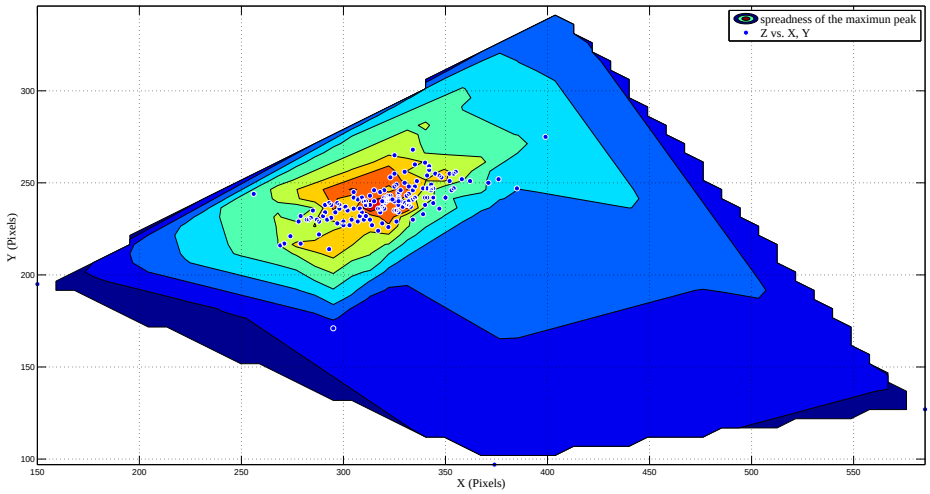


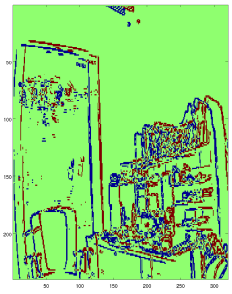
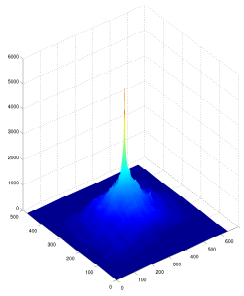
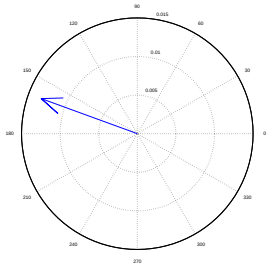
Figure 5.8: Distribution of the global max normalized cross-correlation

5.5 Results

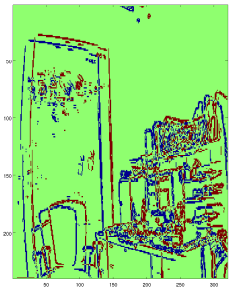
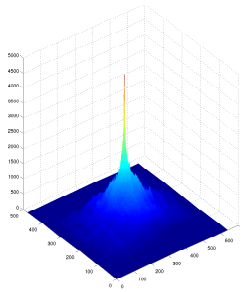
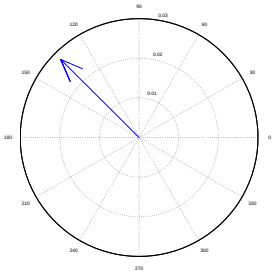
In this section we present the results of the optical flow, cross-correlation, and the ratio of normalized cross-correlation approaches described above. We implemented the algorithms in Matlab and tested them with frame sequences obtained from the citric platform.

5.5.1 Cross-correlation and optical flow

We evaluated various scenarios and motion sequences using healthy individuals which walk and simulate falls. The embedded smart camera was mounted in the arm as is shown in figure 5.2(b).

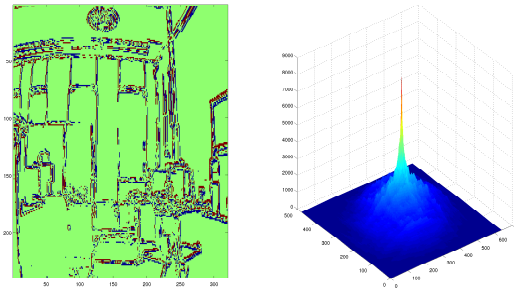
(a) Cross correlation (b) Average Optical Flow

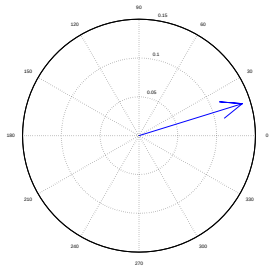
(c) Cross correlation (d) Average Optical Flow

Figure 5.9: Moving to the Left

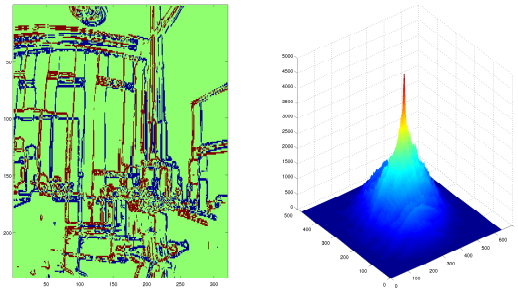
In this experiment the movement sequence is: 1) a person walks normally to the left; 2) a person walks normally to the right; 3) a person falls forward. Figure 5.9 shows the cross correlation and the average optical flow of two consecutive frames where the person is walking normally to the left. Figure 5.10 shows the cross correlation and the average optical flow of two consecutive frames moving to the right. The figures above shows that the magnitude of the average vector does not vary between consecutive frames and its direction keeps to the left or right as expected. Figure 5.11 shows two consecutive frames when the person is falling. Notice that when the person is walking normally to the left or right, the peak of the cross-correlation is centered and above a certain threshold; on the



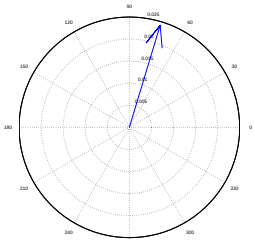
(a) Cross correlation



(b) Average Optical Flow



(c) Cross correlation



(d) Average Optical Flow

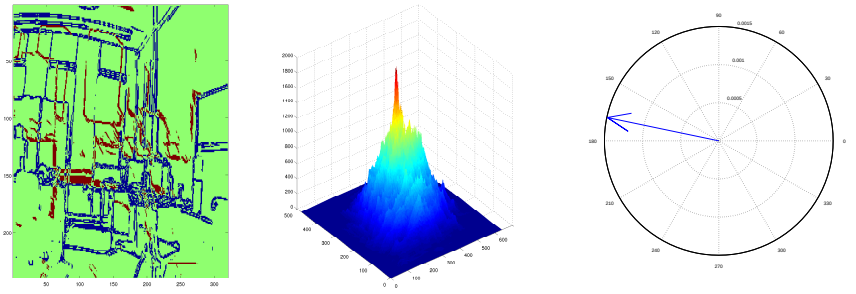
Figure 5.10: Moving to the Right

other hand as is shown in figure 5.11 (a) and (c) when the person is falling the cross-correlation does not have a clear global maxima and the peaks are spread in the whole image.

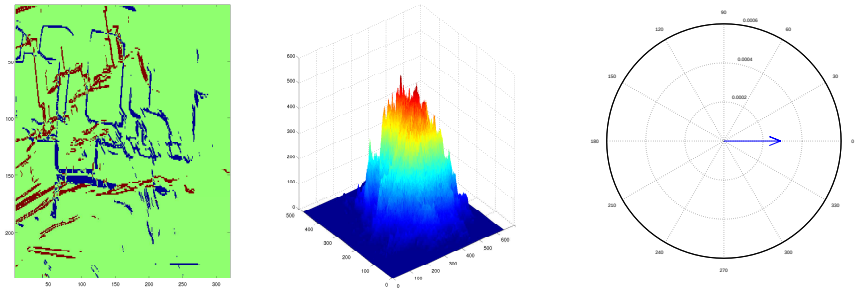
We implemented the cross-correlation in the embedded camera. The problem with this approach is computationally too expensive. It takes approximately 19 seconds to process one frame.

5.5.2 Normalized Cross-correlation ratio

Experimental results of the normalized cross-correlation ratio are presented here. The moving sequences are similar to the previous results, but also more complex



(a) Cross correlation (b) Average Optical Flow



(c) Cross correlation (d) Average Optical Flow

Figure 5.11: Falling

scenarios, such as walking up or down stairs are presented.

Figure 5.12 shows a sequence where a person: 1) walks normally, 2) falls to his/her side and 3) stands up. Fig. 5.13 presents a situation where a person: 1) walks around a chair and 2) sits in a normal manner. In these two sequences we can notice that the peaks of the normalized ratio clearly overpass a threshold which represent a fall. The algorithm does not distinguish between a person falling and standing up because the surrounding conditions are the same and depend how fast the person recovers from a fall. But, a fall and a normal situation, such as a slowly sitting down, can be clearly distinguish.

Fig. 5.14 presents a situation where a person walks down stairs. This particular

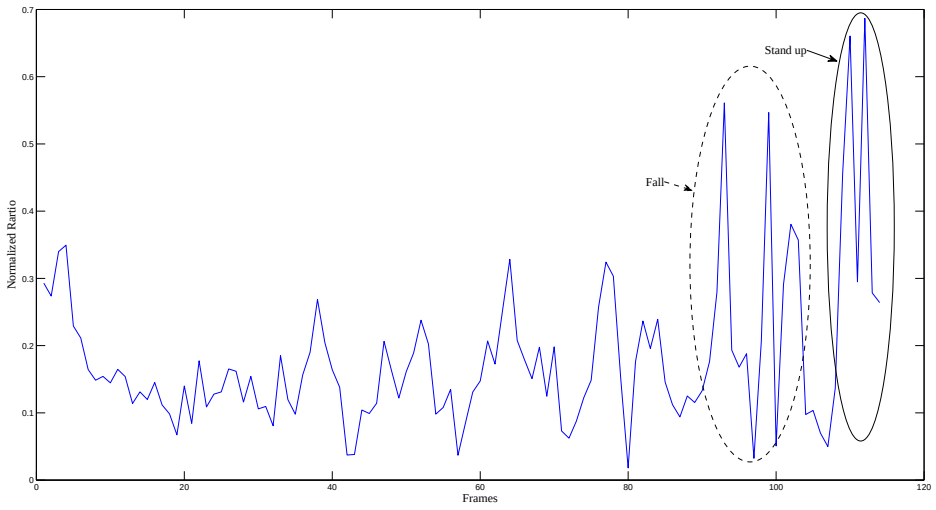


Figure 5.12: Sequence walking-falling

situation is very difficult for the algorithm because in this particular place the walls usually are very close and the algorithm cannot distinguish the edges as clearly. Additionally, these places do not have good illumination.

In this chapter, we tested different approaches to detect falls, but the implementation in a wireless embedded camera it is very difficult due to their complexity. Consequently, a novel approach must be develop which takes in consideration the limited resources of the embedded camera. This approach is properly investigated in chapter 6. Also, a novel algorithm is proposed and implemented in an actual wireless embedded camera.

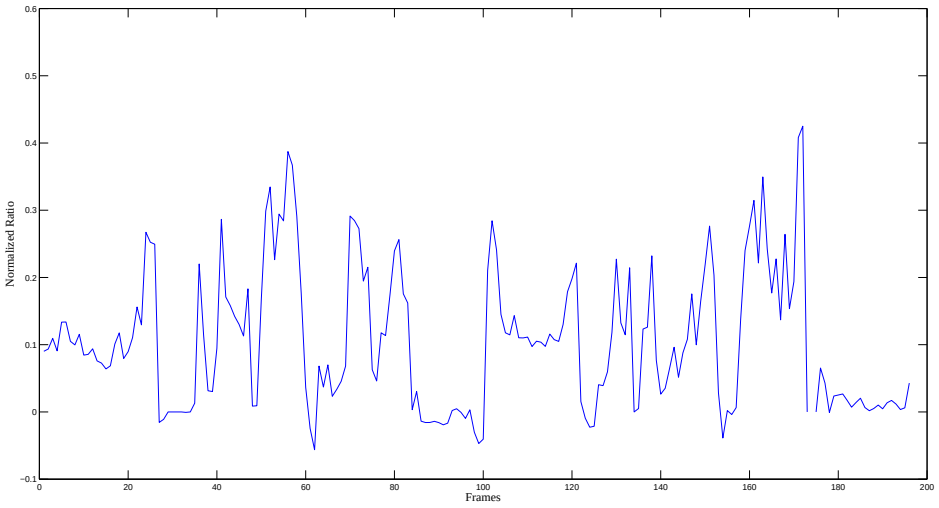


Figure 5.13: Sequence walking-normal sitting

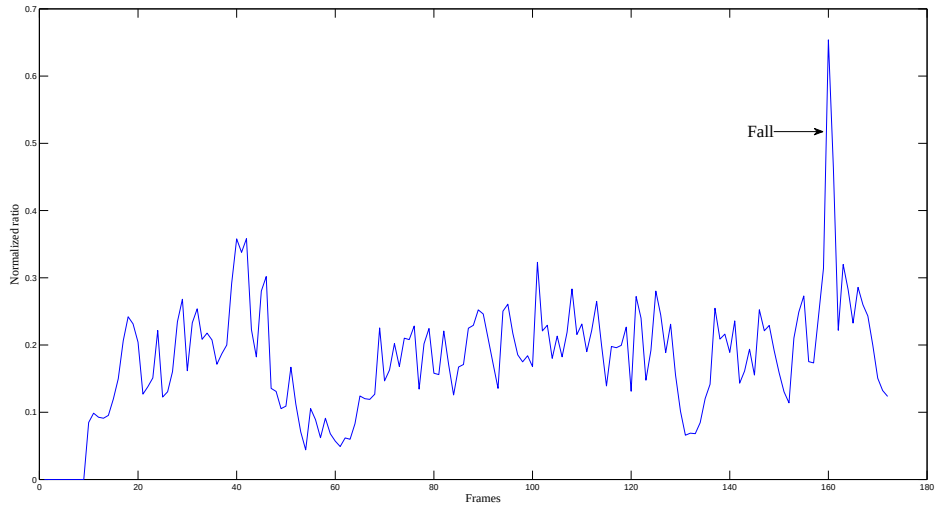


Figure 5.14: Sequence stairs

Chapter 6

Lightweight Fall Detection Algorithm for Wireless Embedded Smart Cameras

6.1 Introduction

In this chapter, we present our method for computing derivatives needed to estimate the user's movement and detect abnormal activity, such as falls. As previously stated, we need to compute the spatial derivatives of an image $I_{m \times n}$ at time n and the temporal derivative between time t and $t + \delta t$. The difference now is that we are going to use both gradients separately, obtain a temporal derivative for each, and use this as a feature. The diagram of our approach is presented in Fig. 6.1.

First, the gradients in the horizontal and vertical direction are calculated for frames n and $n - 1$. Then, the temporal difference between the gradient image is computed. Then, we apply a of moving sum approach to highlight vertical

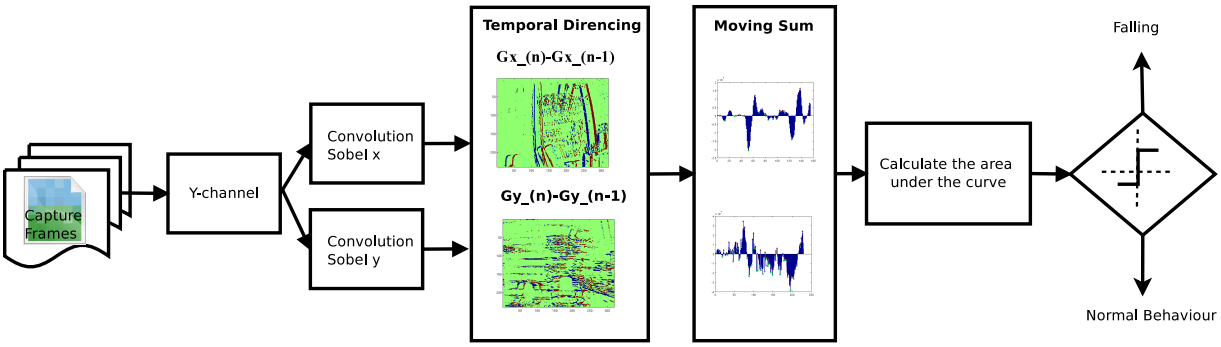


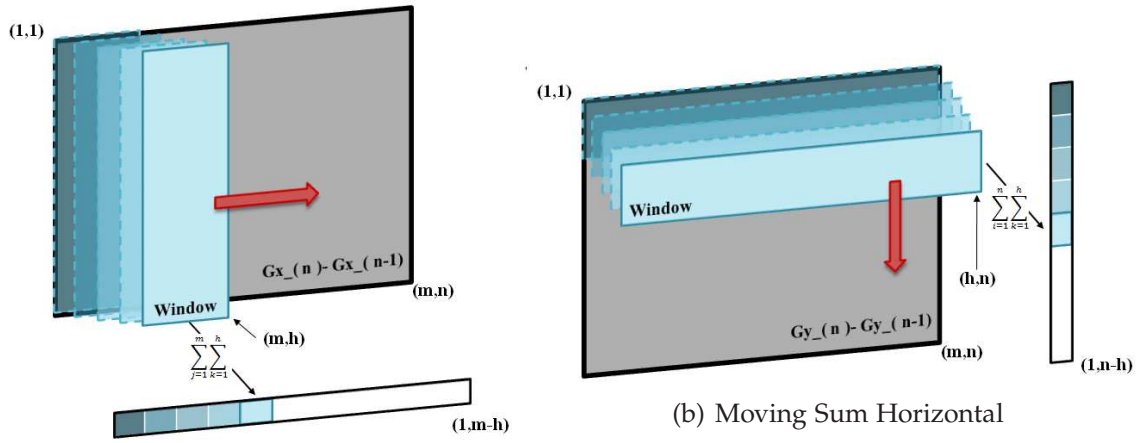
Figure 6.1: Flow Algorithm

and horizontal edges. The result is saved in a vector which is a waveform that represent the edges at time n and $n-1$. If the edges in the previous and in the current frame look similar, then the area under this waveform will be small. The final decision is based on a threshold that is obtained empirically.

6.1.1 Moving sum approach

We compute the gradient of the Y-channel brightness in the horizontal and vertical directions, and form a 2D vector.[71] . We use this information to obtain the horizontal and vertical edges, then we obtain a temporal derivative between them, denoted **T_diff**. Now, we have to determine how different the current frame is compare from the previous frame, knowing that the pixels which do not overlap contain $+1$ or -1 values . To tackle this problem we use a moving sum approach for its good statistical performance. It uses a sliding window of size (m, h) and (h, n) for the vertical and horizontal edges respectively.

A moving sum function, like a moving average, is used to smooth the effects of fluctuations and highlight longer-term trends or cycles. Mathematically, a moving sum is a type of convolution and so it can be viewed as an example of a low-pass



(a) Moving Sum Vertical

(b) Moving Sum Horizontal

Figure 6.2: Moving Sum approach

filter. Statistically, the moving average is optimal for recovering the underlying trend of the time series when the fluctuations about the trend are normally distributed [72]. In this approach we use a rectangular window also known as a Dirichlet window. It takes a chunk of the signal without any other modification, which leads to discontinuities at the endpoints. An advantage of using a moving sum approach is that they are also building blocks from which other filters can be constructed [73, 74].

A moving sum calculation creates a vector **sum**; such, that each element is the sum of the elements in the window of an input **T_diff**. Figure 6.3(a) shows the temporal derivative where the negative and positive values are represented in blue and red, respectively. Four sections are highlighted, regions 1 and 2 contain vertical edges, region 3 contains sparse edges and region 4 shows vertically superimposed edges . Figure 6.3(b) shows the four sections described above, where the edges corresponding to the previous frame are in the negative side of the y axis and the edges of the current frame in the positive side.

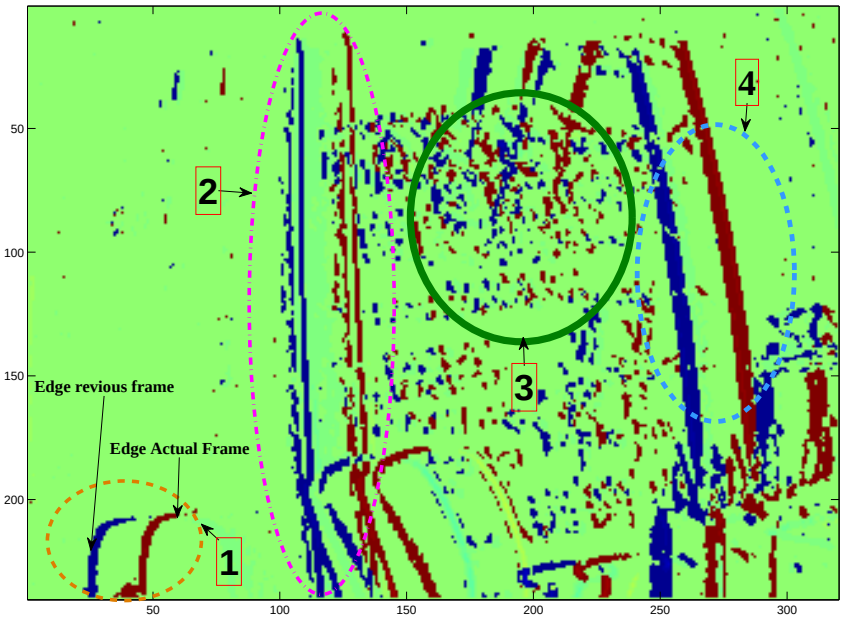
Notice that, edges which are vertically superimposed are clearly visible in region 4, meaning they can be detected and regardless their position they highlight the trend of the edges' motion. Sparse edges are detected as well in region 3, which represent an advantage in difficult scenes, where no strong vertical or horizontal edges are detected. If we assume that the edges from the previous frame are similar to the current edges, the area under this curve will be a low value compared to a frames where a falls occurs. This algorithm is computationally inexpensive and provides useful match estimates. The size of the window was chosen empirically. The size of the window used for the results is **10**. It is large enough to capture the regions with physical motion and still computationally inexpensive.

Let h denoted the window size used in moving sum. The moving sum along the **x** and y axes is computed as follows:

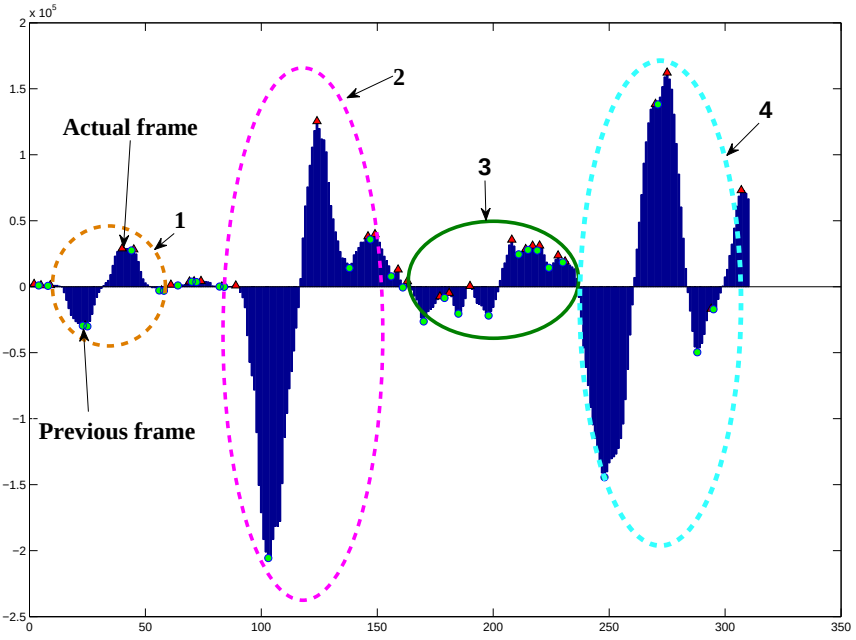
$$\mathbf{SX}_{1\times m-h} = \sum_{j=1}^{m}\sum_{k=1}^{h} T\_diffX(j, i+k) \tag{6.1}$$

$$\mathbf{SY}_{1\times n-h} = \sum_{i=1}^{n}\sum_{k=1}^{h} T\_diffY(j+k, i) \tag{6.2}$$

Figure 6.4 shows results for different window sizes. We tried reducing the window size and making the step size two columns. Smaller window size works well when the camera does not have an inclination. If the camera has an inclination or the temporal derivative has sparse edges, best results are obtained with a higher window size (eg. 10). Fig. 6.5(a) shows a frame where the inclination of the camera is notable; however our approach proves to be effective since the total area under the curve is a low value. Fig. 6.5(b) shows the moving sum for a difficult scene wherein there are many sparse edges.

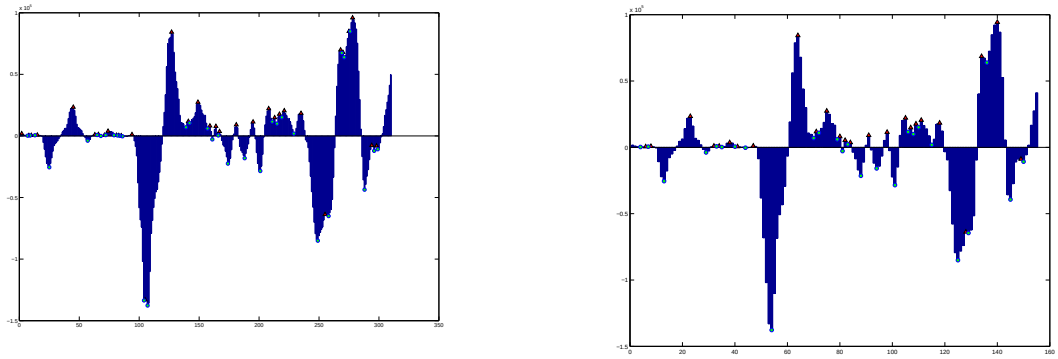


(a) Temporal Derivative



(b) Edges in 2D Moving steps of 1 and window size 10

Figure 6.3: Moving Sum approach



(a) Step=1 and window size=5

(b) Step=2 and window size=5

Figure 6.4: 1D representation of summation vector

After the area is computed, thresholding is apply to determine whether a fall has occurred. We use the following two rules:

- If either the area for the x or y are higher than a threshold a *warning* message is generated .
- If the area for the x and y is greater than a threshold 3 times in the same frame range an *alarm* message is generated

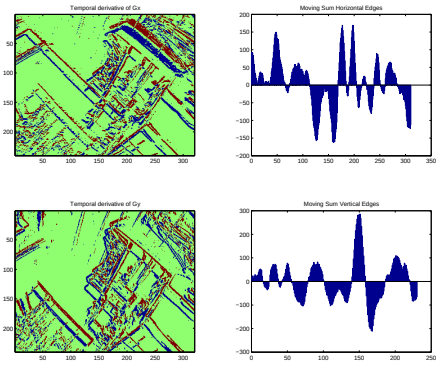
The values for the thresholds were obtained empirically, based on many many experiments where for a simulated fall the average of peaks above the threshold is 3.

6.1.2 Pseudo-code

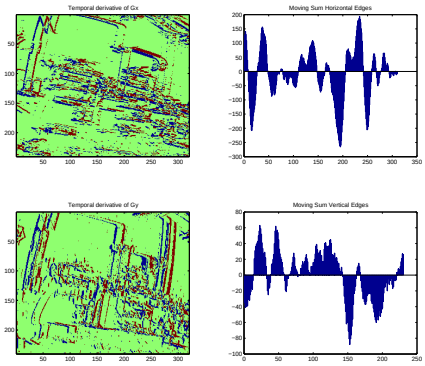
Pseudo-code of our propose method is presented in algorithm 1.

Algorithm 1 Fall Detection

1. **loop**
2. I = Grab frame (YUV,$m \times n$) {color apace, size of the image}
3. **if** First frame **then**
4. Initialization
5. **else**
6. Convolve Y channel with Sobel operators
7. $\mathbf{G}x(n)_{m\times n} = Y_{m\times n} * sobelx_{3\times 3}$
8. $\mathbf{G}y(n)_{m\times n} = Y_{m\times n} * sobely_{3\times 3}$
9. Thresholding
10. $\mathbf{G}x(n)_{m\times n} = (\mathbf{G}x(n) > Threshold)$
11. $\mathbf{G}y(n)_{m\times n} = (\mathbf{G}y(n) > Threshold)$
12. Temporal Differencing
13. $T\_diffX_{m\times n} = \mathbf{G}x(n) - \mathbf{G}x(n-1)$
14. $T\_diffY_{m\times n} = \mathbf{G}y(n) - \mathbf{G}y(n-1)$
15. Compute a vector which contains the moving sum.
16. where: h is the size of the window.
17. **for** $i = 1$ to $n - h$ **do**
18. $SX(i) = \sum_{j=1}^{m} \sum_{k=1}^{h} T\_diffX(j, i+k)$ {Highlight vertical edges}
19. **end for**
20. **for** $j = 1$ to $m - h$ **do**
21. $sumY(j) = \sum_{i=1}^{n} \sum_{k=1}^{h} tmp\_diffX(j+k, i)$ {Highlight horizontal edges}
22. **end for**
23. Calculate the area
24. $area\_x = \sum_{i=1}^{n-h} vect\_sumX(i)$
25. $area\_y = \sum_{i=1}^{m-h} vect\_sumY(i)$
26. Detect falls using thresholding
27. **if** $area\_x > Threshold_{vertical}$ **and** $area\_y > Threshold_{horizontal}$ **then**
28. For more than 2 times in a frame range $\longrightarrow$ **ALARM FALLING**
29. otherwise $\longrightarrow$ warning
30. **else**
31. Normal Behavior
32. **end if**
33. **end if**
34. **end loop**



(a) A scene with camera inclination



(b) Sparse

Figure 6.5: 1D representation of summation vector

6.2 Experimental Results

6.2.1 Camera Setup

We implemented the algorithm on a CITRIC mote, which is a wireless embedded smart camera. It consists of a camera board and a wireless mote. The camera board is composed of a CMOS image sensor, a microprocessor, external memories and other supporting circuits. The image array is capable of operating at up to 30 fps in VGA and lower resolutions. The microprocessor PXA270 is a fixed-

point processor with a maximum speed of 624MHz and 256KB of internal SRAM. An embedded Linux system runs on the camera board. The wireless mote employed is a TelosB mote. The TelosB uses an IEEE 802.15.4-compliant radio. The maximum data rate of the TelosB is 250kbps [29].

The camera was strapped on the arm (as shown in figure 5.2(b)) of a healthy person to test the system functionalities. Data was collected in a text file an then transfer via serial port for post processing. We experiment with different fall scenarios. The algorithm is completely implemented in the wireless embedded smart camera, the post processing using the text file is for debugging and visualization purposes.

6.2.2 Data Collection and Fall Activity Monitoring

Yu et al. [62], established that elderly people and patients are threatened mainly by three scenarios of fall occurrences: fall from sleeping (bed); fall from sitting (chair); fall from walking or standing on the floor. Our approach can be applied to the last two cases. These two classes of fall share some common characteristics; however, they posses different characteristics as well. There is no specific study that shows the characteristics of fall, however few studies present some fall characteristics [113] :

- **The characteristics of fall from walking or standing.**

    1. A fall is a process lasting 1 to 2 seconds, consisting of several sub-actions.
    2. The person stands at the beggining of the fall.
    3. The head lies on the floor in the end of fall process. The head would lie on the floor motionless or with little motion for a while.

4. A person falls roughly in one direction. As a result, both the head and the weight center of the person move approximately in one plane during falling.
5. The head reduces its weight from the standing height to the lying height(lying on the floor). Within portion of this period, the head will fall in a free-fall manner.
6. The lying head is within a circle centered at the foot position of the last standing time.

- **The characteristics of fall from sitting (or chair).**

1. A fall is a process lasting 1 to 3 seconds, consisting of several sub-actions.
2. The person is sitting in the chair at the beggining of the fall.
3. The head reduces its height from the sitting height to the lying height(lying on the floor). Within portion of this period, the head will fall in a free fall manner.
4. The lying body on the floor is nearby the chair.

This information can be helpful to determine a first estimation of the threshold and the frame range. We used these characteristics as a reference to execute our experiments.

We performed a series of experiments to determine a suitable threshold and frame rate under different circumstances. Figure 6.6 shows the case when a person walks randomly and then falls, namely scenario 1. The alarm is generated when the condition is satisfied for both horizontal and vertical movement in the same range frame.

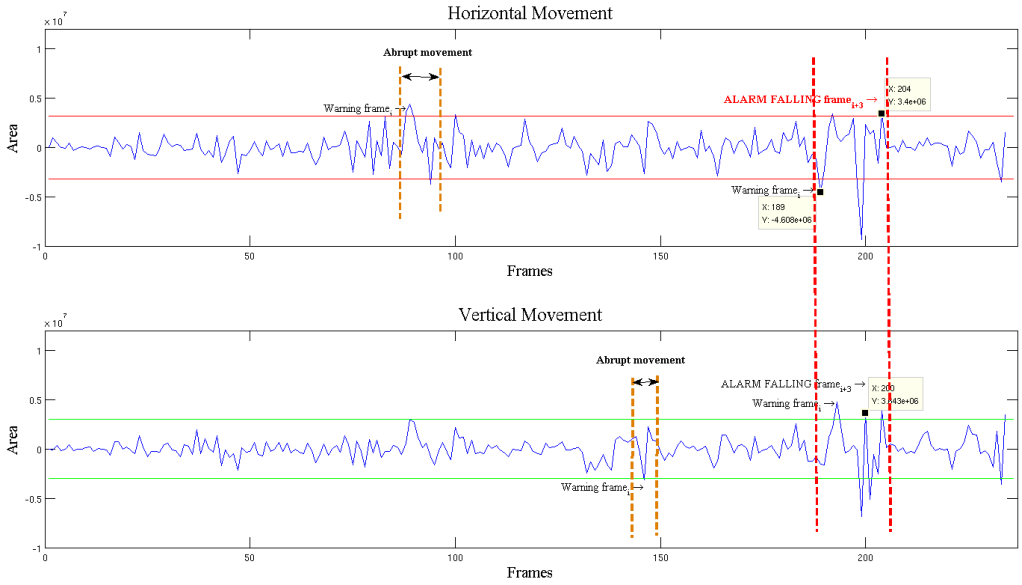


Figure 6.6: Fall Detection scenario 1

Figure 6.7 depicts a series of movements, namely, scenario 2: walk forward - turn left - walk forward - turn right - walk forward- bend to the front - return to straight position - sit gently - stand up gently - keep standing for few seconds and then a fast sit. Figure 6.8 shows a sequence where a person is standing without any movement, then walks forward and backward, falls, lies in the floor for few seconds and then stands up and walks away (scenario 3).

6.2.3 Evaluation

The evaluation criteria are the correct detection of falls, and the correct average time of falling in terms of frames. In the analysis of different scenarios we were partially successful at detecting falls and differentiating them from other conditions that can generated false alarms, such as fast sitting. Our approach was

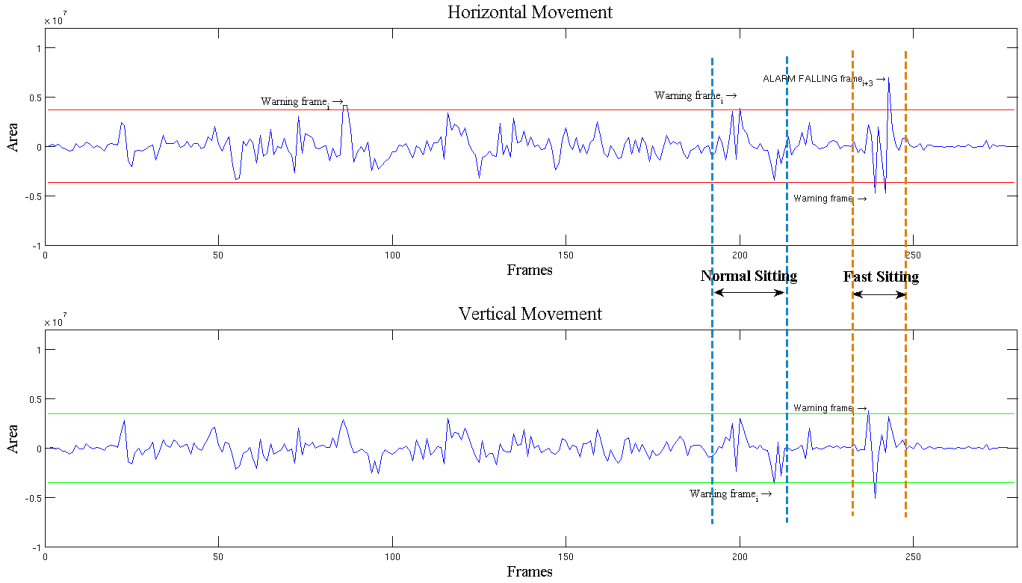


Figure 6.7: Fall detection scenario 2

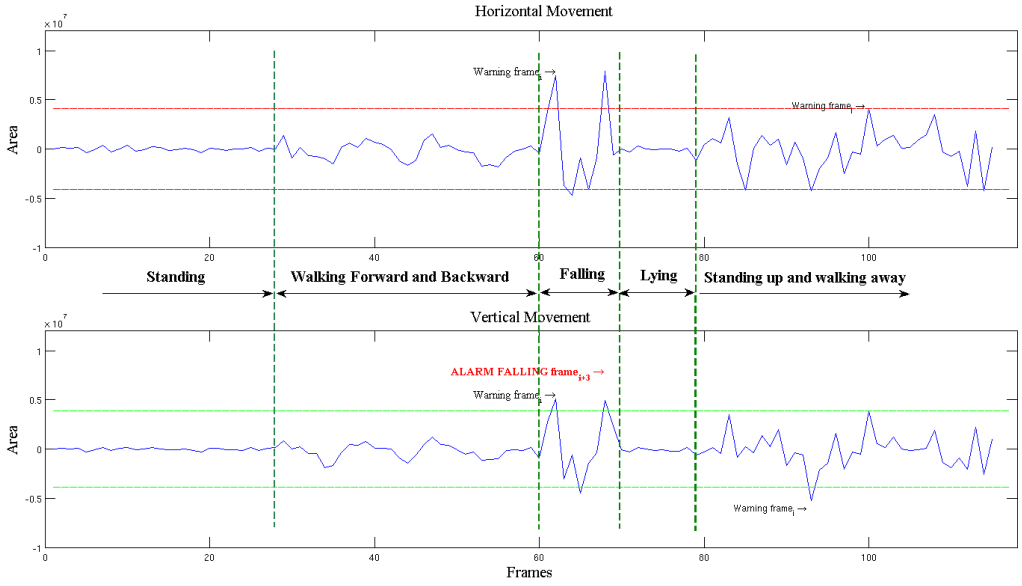


Figure 6.8: Fall detection scenario 3

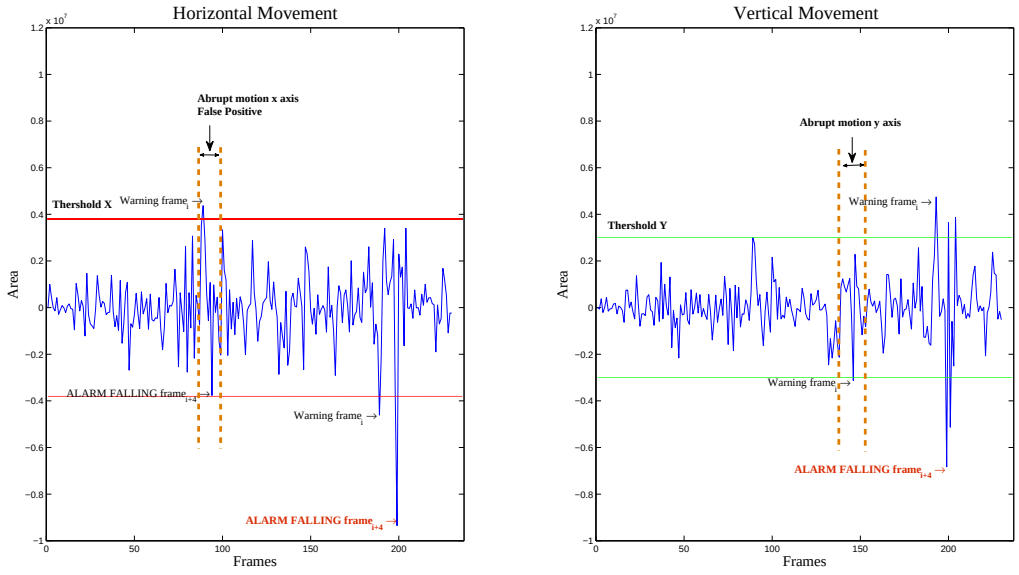


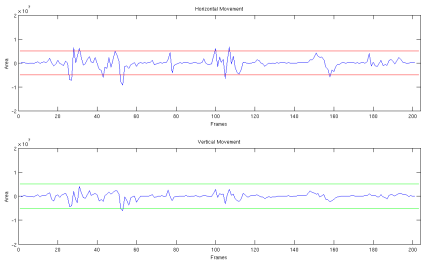
Figure 6.9: Fall Detection using different thresholds

also able to generated warning, when sudden movements were performed. The threshold to declare a fall is approximately 3.7×10^6. Figure 6.9 shows that we can even set different thresholds for the vertical and horizontal motion, we can be a little bit more flexible in the horizontal movement since more abrupt movements occur in this axis. The fall window is around 15 frames.

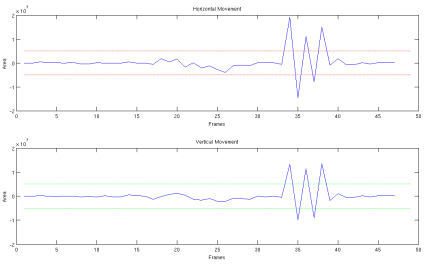
6.2.4 Fall Detection Testing

A series of experiments were carried out wherein different common activities in order to test the reliability of the algorithm. Figure 6.10 shows the results.

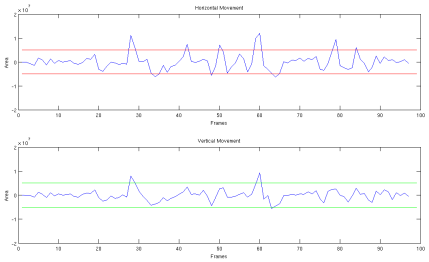
For these experiments two different persons were tested wearing the camera: person A and person B. The location was a house to simulate a realistic scenario. In Figure 6.10(a) person A walks slowly entering the living room from a nar-



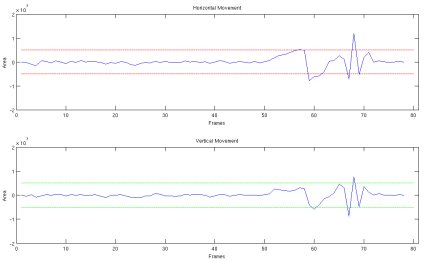
(a) Experiment 1 person A



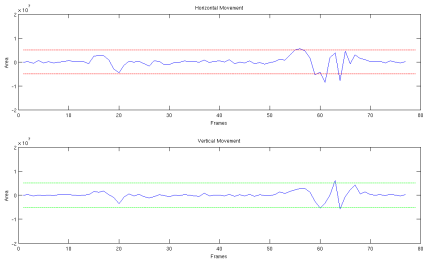
(b) Experiment 2 person A



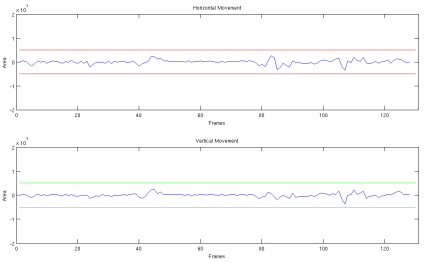
(c) Experiment 3 person B



(d) Experiment 4 person B



(e) Experiment 5 person B



(f) Experiment 6 person A

Figure 6.10: Validation Experiments for fall detection under diverse circumstances and motion using CITRIC motes for

row corridor. The scenario changes in terms of light and background conditions, which makes it a complex scenario. In Figure 6.10(b) person A walks in the living room and then sits fast on a couch. Figure 6.10(c) shows the performance of the algorithm for a scenario wherein person B enters the living room. Then, in figure 6.10(d) and 6.10(e) person A walks and then sits normally and fast respectively. In fig. 6.10(f) person A walks forward and backward then slowly picks something from the floor and then returns to a straight up position in a normal manner. The results presented above were used to detect the threshold. Thus, the algorithm does not generated an alarm for this specific scenarios.

The results presented reveal that the method is effective and accurate. It detect all the falls. The system clearly can make the distinction between normal motion and falls, - even if the environmental conditions change rapidly. The only scenario where the algorithm cannot differentiate between a normal situation and a fall is when the person sits rapidly due to the similarity of the movements.

The time needed to analyze one image composed of actual and previous frame was about 301 milliseconds on a CITRIC mote.

The CITRIC mote perform the following:

- Grabs current frame.
- Applies Sobel operator to detect vertical and horizontal movement.
- Computes temporal differencing between current and previous frame.
- Computes a vector which contains the moving sum.
- Calculates the area.
- Detects falls using thresholding.

- Generates a history file.

Chapter 7

Conclusions

In this work, we first presented a performance analysis for wireless embedded cameras, and then, analyzed the accuracy-latency-tradeoff for different scenarios. In the first scenario, a scalar sensor wakes up the camera mote when it detects motion in the scene. The camera captures a frame, but does not perform any local processing of the image. It transmits the whole image frame in a multi-hop manner to a sink node. In the second scenario, cameras perform local processing to detect foreground objects. The camera transmits only when the size of the detected object satisfies a specified criteria (for instance to detect large vehicles). In addition, the camera does not transmit the whole frame, but only transmits the portion containing the object. In the third scenario, after performing local processing of the image, one camera communicates with another camera in a P2P manner to detect a composite event, and only when the composite event is detected, they will transmit the interesting portion of a frame to the sink. We have presented a detailed quantitative comparison of these three scenarios in terms of the energy consumption and latency when the goal is detecting a composite and semantically high-level event. In addition to providing motivation for and

emphasizing the importance of pushing the high-level decision making to the sensor level, this analysis gives quantitative results in terms of savings in energy.

Also, we have presented a wireless embedded smart camera network for multi-camera tracking. In multi-camera tracking applications, the amount of data exchanged between cameras has an effect on the tracking accuracy, the energy consumption of the camera nodes and the latency. We have provided a detailed quantitative analysis of the accuracy- latency-energy tradeoff for overlapping and non-overlapping camera setups when different-sized data packets are transferred in a wireless manner. The experiments have been performed with an actual wireless embedded smart camera network employing CITRIC motes and performing tracking of objects. Our results show that, for the studied scenarios, deploying four overlapping embedded cameras and communicating less data consumes 23.56 % less energy than using two non-overlapping cameras and exchanging compressed color histograms. In addition, when using cameras with overlapping fields of view, the reliability is very high for uncrowded scenes. The reliability of using only gray-level histogram for non-overlapping cameras decreases especially when there are multiple objects in the scene with similar brightness values. Using all three channels increases the accuracy with the cost of higher delays and higher energy consumption.

Finally, we have presented the development and testing of a lightweight algorithm to perform fall detection for eldercare. Nowadays, we different technologies are suggested to help elderly people in case of emergency. We have proposed a novel method to detect falls, using a wearable wireless embedded camera device. This method analyzes images. It has low computational cost and fast response.

Experimental results illustrate and verify that the system can successfully detect and store fall-related data. The data base provides important information to

analyze long-term trends and patterns needed in the prevention of falls. Other complex fall scenarios are currently under investigation.

The proposed method can differentiate falls from most of the regular mention patterns that may generate false alarms such as bending down. One exception is fast sitting action due to similarities in motion and edge changes

To distinguish these activities, context information (environmental/physiological) can be exploited in future work.

Bibliography

[1] I. Akyildiz, T. Melodia, and K. Chowdhury, "Wireless multimedia sensor networks: Applications and testbeds," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1588 –1605, 2008. 1.1, 2.1, 2.5.1, 2.5.2, 3.1, 4.1, 5.1, 5.3

[2] M. Imran, K. Khursheed, M. O'Nils, and N. Lawal, "Exploration of target architecture for a wireless camera based sensor node," in *NORCHIP, 2010*, 2010, pp. 1 –4. 1.1, 2.1

[3] B. Harjito and S. Han, "Wireless multimedia sensor networks applications and security challenges," in *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on*, 2010, pp. 842 – 846. 2.1, 2.2, 2.6.6

[4] N. Kahar, R. Ahmad, Z. Hussin, and A. Rosli, "Embedded smart camera performance analysis," in *Computer Engineering and Technology, 2009. ICCET '09. International Conference on*, vol. 2, 2009, pp. 79 –83. 2.1

[5] W. Wolf, B. Ozer, and T. Lv, "Smart cameras as embedded systems," *Computer*, vol. 35, no. 9, pp. 48 – 53, Sep. 2002. 2.1

[6] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach, "Distributed embedded smart cameras for surveillance applications," *Computer*, vol. 39, no. 2, pp. 68 – 75, 2006. 2.1

[7] B. Rinner and W. Wolf, "A Bright Future for Distributed Smart Cameras," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1562–1564, Oct. 2008. 2.1, 2.2

[8] C. Wu, H. Aghajan, and R. Kleihorst, "Real-Time Human Posture Reconstruction in Wireless Smart Camera Networks," *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, pp. 321–331, Apr. 2008. 2.1, 2.2

[9] T. Winkler and B. Rinner, "Pervasive smart camera networks exploiting heterogeneous wireless channels," *2009 IEEE International Conference on Pervasive Computing and Communications*, pp. 1–4, Mar. 2009. 2.1, 2.2

[10] P. Wang, R. Dai, and I. Akyildiz, "Collaborative data compression using clustered source coding for wireless multimedia sensor networks," in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1 –9. 2.1

[11] M. Kushwaha and X. Koutsoukos, "Collaborative target tracking using multiple visual features in smart camera networks," in *Information Fusion (FUSION), 2010 13th Conference on*, 2010, pp. 1 –8. 2.1

[12] G. Benet, J. Simo and, G. Andreu-Garci anda, J. Rosell, and J. Sa andnchez, "Embedded low-level video processing for surveillance purposes," in *Human System Interactions (HSI), 2010 3rd Conference on*, May 2010, pp. 779 –786. 2.1

[13] K. S. Cauligi Raghavendra and T. Znati, *Wireless Sensor Networks*. Springer, 2006. 2.2

[14] A. Sankaranarayanan, A. Veeraraghavan, and R. Chellappa, "Object detection, tracking and recognition for multiple smart cameras," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1606 –1624, 2008. 2.2

[15] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer Networks*, vol. 51, pp. 921–960, 2007. 2.2, 3.1

[16] S. Soro and W. Heinzelman, "A Survey of Visual Sensor Networks," *Advances in Multimedia*, vol. 2009, pp. 1–22, 2009. 2.2, 2.5.2

[17] S. Yu, "Smart cameras for machine vision," in *Smart Cameras*, A. N. Belbachir, Ed. Springer US, 2010, pp. 283–303. 2.2, 2.3.1, 2.3.2, 2.6.4

[18] R. J. Radke, "A survey of distributed computer vision algorithms," in *Aghajan (Eds.), Handbook of Ambient Intelligence and Smart Environments*. Springer, 2008. 2.2

[19] H. Aghajan and A. Cavallaro, *"Multi-Camera Networks Principles and Applications"*, 1st ed. BurLington, MA 01803: Elsevier, 2009. 2.2, 2.3.2, 2.4

[20] B. Rinner and W. Wolf, "Towards pervasive smart camera networks," *Computer Engineering*, no. July 2009, 2009. 2.2, 2.3.1, 2.3.2

[21] S. Fleck and W. Strasser, "Smart camera based monitoring system and its application to assisted living," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1698 –1714, 2008. 2.2, 5.1, 5.3

[22] C. D. Kidd, R. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. Macintyre, E. Mynatt, T. E. Starner, and W. Newstetter, "The aware home: A living laboratory for ubiquitous computing research," *Ieee Spectrum*, pp. 191–198, 1999. 2.2

[23] T. Winkler and B. Rinner, "Securing embedded smart cameras with trusted computing," *CResearch Article*, 2010. 2.2

[24] S. Fleck and W. Straer, "Privacy sensitive surveillance for assisted living a smart camera approach," in *Handbook of Ambient Intelligence and Smart Environments*, H. Nakashima, H. Aghajan, and J. C. Augusto, Eds. Springer US, 2010, pp. 985–1014. 2.2

[25] F. D. Real and F. Berry, "Smart cameras: Technologies and applications," in *Smart Cameras*, A. N. Belbachir, Ed. Springer US, 2010, pp. 35–50. 2.2, 2.3, 2.3.2

[26] S. Hadim and N. Mohamed, "Middleware: middleware challenges and approaches for wireless sensor networks," *Distributed Systems Online, IEEE*, vol. 7, no. 3, p. 1, march 2006. 2.2, 2.4

[27] T. Camilo, R. Oscar, and L. Carlos, "Biomedical signal monitoring using wireless sensor networks," *2009 IEEE Latin-American Conference on Communications*, pp. 1–6, Sep. 2009. 2.3.3, 5.2

[28] M. M. Molla and S. I. Ahamed, "A survey of middleware for sensor network and challenges," in *Proceedings of the 2006 International Conference Workshops on Parallel Processing*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 223–228. 2.4

[29] P. W.-C. Chen, P. Ahammad, C. Boyer, S.-I. Huang, L. Lin, E. J. Lobaton, M. L. Meingast, S. Oh, S. Wang, P. Yan, A. Yang, C. Yeo, L.-C. Chang, D. Tygar, and S. S. Sastry, "Citric: A low-bandwidth wireless camera network platform," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-50, May 2008. 2.5.2, 3.1, 5.3, 6.2.1

[30] G. Diraco, A. Leone, and P. Siciliano, "An Active Vision System for Fall Detection and Posture Recognition in Elderly Healthcare," *Image (Rochester, N.Y.)*, 2010. 2.6.5, 5.1, 5.2

[31] Y. Liu and C. Pomalaza-Raez, "Detection of body posture with low-cost CMOS camera systems for applications in medical care," *2009 IEEE International Conference on Electro/Information Technology*, pp. 301–306, Jun. 2009. 2.6.5, 5.1, 5.2

[32] B. Rinner and W. Wolf, "An introduction to distributed smart cameras," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1565 –1575, 2008. 3.1

[33] D.-U. Lee, H. Kim, S. Tu, M. Rahimi, D. Estrin, and J. Villasenor, "Energy-optimized image communication on resource-constrained sensor platforms," in *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, april 2007, pp. 216 –225. 4.1

[34] H. Wu and A. A. Abouzeid, "Energy efficient distributed image compression in resource-constrained multihop wireless networks," *Comput. Commun.*, vol. 28, pp. 1658–1668, September 2005. 4.1

[35] P. Kulkarni, D. Ganesan, P. J. Shenoy, and Q. Lu, "Senseye: a multi-tier camera sensor network." in *ACM Multimedia'05*, 2005, pp. 229–238. 4.1

[36] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan, "Mesheye: A hybrid-resolution smart camera mote for applications in distributed intelligent surveillance," in *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, 2007, pp. 360 –369. 4.1

[37] P. Skraba and L. Guibas, "Energy efficient intrusion detection in camera sensor networks," pp. 309,323, 2007. 4.1

[38] C. Margi, V. Petkov, K. Obraczka, and R. Manduchi, "Characterizing energy consumption in a visual sensor network testbed," in *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006. 2nd International Conference on*, 0-0 2006, pp. 8 pp. –339. 4.1

[39] T. Ko, Z. Charbiwala, S. Ahmadian, M. Rahimi, M. Srivastava, S. Soatto, and D. Estrin, "Exploring tradeoffs in accuracy, energy and latency of scale invariant feature transform in wireless camera networks," in *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, sept. 2007, pp. 313 –320. 4.1

[40] S. Velipasalar, J. Schlessman, C.-Y. Chen, W. Wolf, and J. Singh, "A scalable clustered camera system for multiple object tracking," in *EURASIP Journal on Image and Video Processing*, no. 542808, 2008, pp. 277 –280. 1

[41] M. Casares, S. Velipasalar, and A. Pinto, "Light-weight salient foreground detection for embedded smart cameras," *Computer Vision and Image Understanding*, vol. 114, no. 11, pp. 1223 – 1237, 2010, special issue on Embedded Vision. 4.3.2.1

[42] Y. Wang, S. Velipasalar, and M. Casares, "Cooperative object tracking and composite event detection with wireless embedded smart cameras," *Image Processing, IEEE Transactions on*, vol. 19, no. 10, pp. 2614 –2633, oct. 2010. 4.3.2.1

[43] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2, 2000, pp. 142 –149 vol.2. 4.3.2.1

[44] BBC-News. (2010, Jun) "£1 billion cost of elderly falls". [Online]. Available: http://news.bbc.co.uk/2/hi/health/3167005.stm 5.1

[45] ——. (2003, Aug) "falls amongst the elderly cost the nhs millions daily". [Online]. Available: http://www.bbc.co.uk/news/10353642 5.1

[46] D. of Health & Human Services. (2008, September) "costs of falls among older adults". [Online]. Available: http://www.cdc.gov/ncipc/factsheets/fallcost.htm 5.1

[47] H. Huo, Y. Xu, H. Yan, S. Mubeen, and H. Zhang, "An Elderly Health Care System Using Wireless Sensor Networks at Home," *2009 Third International Conference on Sensor Technologies and Applications*, pp. 158–163, Jun. 2009. 5.1

[48] S. Nourizadeh, C. Deroussent, Y. Q. Song, and J. P. Thomesse, "Medical and Home Automation Sensor Networks for Senior Citizens Telehomecare," *2009 IEEE International Conference on Communications Workshops*, pp. 1–5, Jun. 2009. 5.1

[49] N. Raveendranathan, S. Galzarano, V. Loseu, R. Gravina, R. Giannantonio, M. Sgroi, R. Jafari, and G. Fortino, "From Modeling to Implementation of Virtual Sensors in Body Sensor Networks," *Signal Processing*, no. c, 2011. 5.1

[50] P. Zhang and K. Zhang, "A Remote Sleep-monitoring Medical Alarm System," *Control*, vol. 2272, no. 1, pp. 2–5, 2010. 5.1

[51] S. Patel, K. Lorincz, R. Hughes, N. Huggins, J. Growdon, D. Standaert, M. Akay, J. Dy, M. Welsh, and P. Bonato, "Monitoring motor fluctuations in patients with Parkinson's disease using wearable sensors." *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 13, no. 6, pp. 864–73, Nov. 2009. 5.1

[52] X. Yu, "Approaches and principles of fall detection for elderly and patient," in *e-health Networking, Applications and Services, 2008. HealthCom 2008. 10th International Conference on*, july 2008, pp. 42 –47. 5.2

[53] V. Moser-siegmeth, S. Pantelopoulos, P. Sakka, and P. Moraitis, "A Novel Architecture and Process for Ambient Assisted Living - The HERA approach," *Human-Computer Interaction*, 2010. 5.2

[54] M. Grassi, A. Lombardi, G. Rescio, M. Ferri, and P. Malcovati, "An Integrated System for People Fall-Detection with Data Fusion Capabilities Based on 3D ToF Camera and Wireless Accelerometer," *Electrical Engineering*, pp. 1016–1019, 2010. 5.2

[55] T. Teixeira, D. Jung, G. Dublon, and A. Savvides, "Pem-id: Identifying people by gait-matching using cameras and wearable accelerometers," in *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, 30 2009-sept. 2 2009, pp. 1 –8. 5.2

[56] A. Dinh, D. Teng, L. Chen, Y. Shi, C. McCrosky, J. Basran, V. Del Bello-Hass, S. Ko, A. Ralhan, D. Williams, N. Windels, and A. Choudhury, "A fall detection and near-fall data collection system," in *Microsystems and Nanoelectronics Research Conference, 2008. MNRC 2008. 1st*, oct. 2008, pp. 117 –120. 5.2, 5.3

[57] T. Gao, T. Massey, M. Sarrafzadeh, L. Selavo, and M. Welsh, "Participatory user centered design techniques for a large scale ad-hoc health information system," *Proceedings of the 1st ACM SIGMOBILE international workshop on Systems and networking support for healthcare and assisted living environments - HealthNet '07*, p. 43, 2007. 5.2

[58] H. Yan, H. Huo, I. Y. Xu, and M. Gidlund, "Wireless Sensor Network Based E-Health System Implementation and Experimental Results," vol. 56, no. 4, pp. 2288–2295, 2010. 5.2

[59] O. Chipara, C. Lu, T. C. Bailey, and G.-c. Roman, "Reliable Clinical Monitoring using Wireless Sensor Networks : Experiences in a Step-down Hospital Unit," *Components*, 2010. 5.2

[60] M. Rantz, M. Skubic, and S. Miller, "Using sensor technology to augment traditional healthcare," in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, sept. 2009, pp. 6159 –6162. 5.2

[61] E. Stone and M. Skubic, "Silhouette classification using pixel and voxel features for improved elder monitoring in dynamic environments," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, march 2011, pp. 655 –661. 5.2

[62] X. Yu, "Approaches and principles of fall detection for elderly and patient," in *e-health Networking, Applications and Services, 2008. HealthCom 2008. 10th International Conference on*, july 2008, pp. 42 –47. 5.2, 6.2.2

[63] E. Culurciello, P. Lichtsteiner, and T. Delbruck, "Fall detection using an address-event temporal contrast vision sensor," in *2008 IEEE International Symposium on Circuits and Systems*. Ieee, May 2008, pp. 424–427. 5.2

[64] A. Williams, D. Ganesan, and A. Hanson, "Aging in place: fall detection and localization in a distributed smart camera network," in *Proceedings of the 15th International Conference on Multimedia*, 2007. 5.2

[65] C. Wu, H. Aghajan, and R. Kleihorst, "Real-time human posture reconstruction in wireless smart camera networks," in *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, april 2008, pp. 321 –331. 5.2

[66] S. Fleck and W. Straß er, "Towards Secure and Privacy Sensitive Surveillance," *System*, pp. 126–132, 2010. 5.3

[67] Q. Li, J. A. Stankovic, M. A. Hanson, A. T. Barth, J. Lach, and G. Zhou, "Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information," in *2009 International Conference on Body Sensor Networks*, vol. 9, 2009, p. 138–143. 5.3

[68] Y. Choi, A. S. Ralhan, and S. Ko, "A study on machine learning algorithms for fall detection and movement classification," in *Information Science and Applications (ICISA), 2011 International Conference on*, april 2011, pp. 1 –8. 5.3

[69] Wikipedia. (2011, Apr) "sobel operator the free encyclopedia". [Online]. Available: hhttp://en.wikipedia.org/wiki/Sobeloperator 5.4.1

[70] A. Haro, K. Mori, T. Capin, and S. Wilkinson, "Mobile camera-based user interaction," Tech. Rep., Jan 2009. 5.4.1

[71] Wikipedia. (2011, Jun) "edge detection wikipedia the free encyclopedia". [Online]. Available: http://en.wikipedia.org/wiki/Edgedetection 6.1.1

[72] ——. (2011, Jun) "moving average". [Online]. Available: http://en.wikipedia.org/wiki/Movingaverage 6.1.1

[73] K.-L. Chung and W.-M. Yan, "Fast vectorization for calculating a moving sum," *Computers, IEEE Transactions on*, vol. 44, no. 11, pp. 1335 –1337, nov 1995. 6.1.1

[74] S. Kar, K. G. Mehrotra, and P. Varshney, "Average run length of two-span moving sum algorithms," Electrical Engineering and Computer Science, Syracuse University, Tech. Rep., Mar 2010. 6.1.1

[75] R. Dai, P. Wang, and I. Akyildiz, "Correlation-aware qos routing for wireless video sensor networks," in *GLOBECOM 2010, 2010 IEEE Global Telecommunications Conference*, 2010, pp. 1 –5.

[76] R. Dai and I. Akyildiz, "Joint effect of multiple correlated cameras in wireless multimedia sensor networks," in *Communications, 2009. ICC '09. IEEE International Conference on*, 2009, pp. 1 –5.

[77] ——, "A spatial correlation model for visual information in wireless multimedia sensor networks," *Multimedia, IEEE Transactions on*, vol. 11, no. 6, pp. 1148 –1159, 2009.

[78] R. Dai, P. Wang, and I. Akyildiz, "Correlation-aware qos routing for wireless video sensor networks," in *GLOBECOM 2010, 2010 IEEE Global Telecommunications Conference*, 2010, pp. 1 –5.

[79] T. Melodia and I. Akyildiz, "Cross-layer quality of service support for uwb wireless multimedia sensor networks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, 2008, pp. 2038 –2046.

[80] T. Winkler and B. Rinner, "Power aware communication in wireless pervasive smart camera networks," *2009 International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pp. 283–288, Dec. 2009.

[81] K. Khursheed, M. Imran, M. O. Nils, and N. Lawal, "Exploration of Local and Central Processing for a Wireless Camera Based Sensor Node," *Components*, pp. 147–150, 2010.

[82] W. Schriebl, T. Winkler, A. Starzacher, and B. Rinner, "A pervasive smart camera network architecture applied for multi-camera object classification," in *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, 30 2009-sept. 2 2009, pp. 1 –8.

[83] S. Yu and F. Dias, "Smart cameras: Fundamentals and classification," in *Smart Cameras*, A. N. Belbachir, Ed. Springer US, 2010, pp. 19–34.

[84] R. Holman, J. Stanley, and T. Ozkan-Haller, "Applying video sensor networks to nearshore environment monitoring," *Pervasive Computing, IEEE*, vol. 2, no. 4, pp. 14 – 21, oct.-dec. 2003.

[85] T. Teixeira and A. Savvides, "Lightweight people counting and localizing in indoor spaces using camera sensor nodes," in *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, sept. 2007, pp. 36 –43.

[86] G. Catalano, A. Gallace, B. Kim, F. S. Sergio Pedro, and B. Kim, "Optical flow," Tech. Rep., Mar 2009.

[87] G. Kambourakis, E. Klaoudatou, and S. Gritzalis, "Securing medical sensor environments: The codeblue framework case," in *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, april 2007, pp. 637 –643.

[88] A. M. Tabar, A. Keshavarz, and H. Aghajan, "Smart home care network using sensor fusion and distributed vision-based reasoning," in *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, ser. VSSN '06. New York, NY, USA: ACM, 2006, pp. 145–154.

[89] H. Aghajan, J. C. Augusto, C. Wu, P. McCullagh, and J.-A. Walkden, "Distributed vision-based accident management for assisted living," in *Proceedings of the 5th international conference on Smart homes and health telematics*, ser. ICOST'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 196–205.

[90] C. Doukas, I. Maglogiannis, A. Rouskas, and A. Pneumatikakis, "Emergency incidents detection in assisted living environments utilizing sound and visual perceptual components," in *Proceedings of the 2nd International Conference on PErvasive Technologies Related to Assistive Environments*, ser. PETRA '09. New York, NY, USA: ACM, 2009, pp. 14:1–14:7.

[91] T. Gao, C. Pesto, L. Selavo, Y. Chen, J. Ko, J. Lim, A. Terzis, A. Watt, J. Jeng, B.-r. Chen, K. Lorincz, and M. Welsh, "Wireless Medical Sensor Networks in Emergency Response: Implementation and Pilot Results," *2008 IEEE Conference on Technologies for Homeland Security*, pp. 187–192, May 2008.

[92] A. Lymberis, "Smart wearables for remote health monitoring, from prevention to rehabilitation: current R&D, future challenges," *4th International IEEE EMBS Special Topic Conference on Information Technology Applications in Biomedicine, 2003.*, pp. 272–275, 2003.

[93] C. Rougier and J. Meunier, "Demo : Fall Detection Using 3D Head Trajectory Extracted From a Single Camera Video Sequence," *International Journal*, pp. 2–3, 2007.

[94] D. G. Lorenzo Valeri and P. Jansen, "Business models for ehealth," February 2010.

[95] S. Scott and A. Kovac, "WIRELESS TEMPERATURE MICROSENSORS INTEGRATED ON BEARINGS FOR HEALTH MONITORING APPLICAIONS," pp. 660–663, 2011.

[96] E. Auvinet, F. Multon, Saint-Arnaud, J. Rousseau, and J. Meunier, "Fall detection with multiple cameras: An occlusion-resistant method based on 3D silhouette vertical distribution." *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, pp. 1–36, Oct. 2010.

[97] V. Gupta, P. Udupi, and A. Poursohi, "Early lessons from building Sensor.Network: an open data exchange for the web of things," *2010 8th IEEE In-*

ternational Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 738–744, Mar. 2010.

[98] M. Hossien, Y. Moghaddam, and D. Adjeroh, "A Novel Congestion Control Protocol for Vital Signs Monitoring in Wireless Biomedical Sensor Networks," *Communications Society*, pp. 0–5, 2010.

[99] A. Leone, G. Diraco, and P. Siciliano, "An Automated Active Vision System for Fall Detection and Posture Analysis in Ambient Assisted Living Applications," *October*, pp. 2301–2306, 2010.

[100] T. Winkler and B. Rinner, "TrustCAM: Security and Privacy-Protection for an Embedded Smart Camera Based on Trusted Computing," *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 593–600, Aug. 2010.

[101] A. O. Chipara, C. Lu, T. C. Bailey, G.-c. Roman, and O. Chipara, "Reliable Patient Monitoring : A Clinical Study in a Step-down Hospital Unit Reliable Patient Monitoring : A Clinical Study in a Step-down Hospital Unit," *System*, no. 314, 2009.

[102] A. O. Chipara, C. Brooks, S. Bhattacharya, C. Lu, R. Chamberlain, and O. Chipara, "Reliable Data Collection from Mobile Users for Real-Time Clinical Monitoring Reliable Data Collection from Mobile Users for Real-Time Clinical Monitoring," *Design*, no. 314, 2008.

[103] M. Clarke, J. Fursse, and R. W. Jones, "Early experiences of the use of remote patient monitoring for the long term management of chronic disease." *Conference proceedings : ... Annual International Conference of the IEEE Engineer-*

ing in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference, vol. 2008, pp. 5863–6, Jan. 2008.

[104] R. Mitra, B. Joshi, and A. Mukherjee, "Fault-tolerant wearable computing system architecture for self-health management," *2008 IEEE International Conference on Electro/Information Technology*, pp. 349–354, May 2008.

[105] R. Rashid, S. H. S. Arifin, M. R. A. Rahim, M. A. Sarijari, and N. H. Mahalin, "Home healthcare via wireless biomedical sensor network," *2008 IEEE International RF and Microwave Conference*, pp. 511–514, Dec. 2008.

[106] B. Rinner and W. Wolf, "An Introduction to Distributed Smart Cameras," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1565–1575, Oct. 2008.

[107] H. Aghajan, "Real-Time Human Pose Estimation: A Case Study in Algorithm Design for Smart Camera Networks," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1715–1732, Oct. 2008.

[108] C. Wu, H. Aghajan, and R. Kleihorst, "Real-Time Human Posture Reconstruction in Wireless Smart Camera Networks," *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, pp. 321–331, Apr. 2008.

[109] D. Xie, T. Yan, D. Ganesan, and A. Hanson, "Design and Implementation of a Dual-Camera Wireless Sensor Network for Object Retrieval," *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, pp. 469–480, Apr. 2008.

[110] T. Conference, "Wireless Sensor," *Most*, no. May, pp. 1–6, 2007.

[111] O. Gama, C. Figueiredo, P. Carvalho, and P. M. Mendes, "Towards a Reconfigurable Wireless Sensor Network for Biomedical Applications," pp. 490–495, 2007.

[112] S. Hengstler and H. Aghajan, "Application-Oriented Design of Smart Camera Networks," *2007 First ACM/IEEE International Conference on Distributed Smart Cameras*, pp. 12–19, Sep. 2007.

[113] P. Leijdekkers, V. Gay, and E. Lawrence, "Smart Homecare System for Health Tele-monitoring," *First International Conference on the Digital Society (ICDS'07)*, pp. 3–3, 2007. 6.2.2

[114] T. Massey, F. Dabiri, R. Jafari, H. Noshadi, P. Brisk, W. Kaiser, and M. Sarrafzadeh, "Towards Reconfigurable Embedded Medical Systems," *2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPnP 2007)*, pp. 178–180, Jun. 2007.

[115] H. Na, F. Qin, and D. Wright, "A Smart Vision Sensor for Detecting Risk Factors of a Toddler's Fall in a Home Environment," *2007 IEEE International Conference on Networking, Sensing and Control*, no. April, pp. 656–661, 2007.

[116] M. S. Wegmueller, M. Hediger, T. Kaufmann, F. Buergin, and W. Fichtner, "Wireless Implant Communications for Biomedical Monitoring Sensor Network," *2007 IEEE International Symposium on Circuits and Systems*, pp. 809–812, May 2007.

[117] K. Arshak and E. Jafer, "Wireless Ultra-Low Power Smart Data Acquisition System for Pressure Sensing in Medical Application," *2006 25th International Conference on Microelectronics*, no. Miel, pp. 212–219, 2006.

[118] T. Kijewski-Correa, M. Haenggi, and P. Antsaklis, "Wireless Sensor Networks for Structural Health Monitoring: A Multi-Scale Approach," *17th Analysis and Computation Specialty Conference*, pp. 5–5, 2006.

[119] S.-G. Miaou, "A Customized Human Fall Detection System Using Omni-Camera Images and Personal Information," *1st Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare, 2006. D2H2.*, pp. 39–42, 2006.

[120] Prentza, S. Maglavera, and N. Maglaveras, "Quality Healthcare Management and Well-Being through INTERLIFE Services: New Processes and Business Models," *1st Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare, 2006. D2H2.*, pp. 109–112, 2006.

[121] A. M. Tabar, A. Keshavarz, and H. Aghajan, "Smart home care network using sensor fusion and distributed vision-based reasoning," *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks - VSSN '06*, p. 145, 2006.

[122] Reeves, J. Ng, S. J. Brown, and N. Barnes, "Remotely Supporting Care Provision for Older Adults," *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*, pp. 117–122, 2006.